



INSTITUTO POLITÉCNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO

# Práctica 5 - Administrador de procesos en Linux y Windows (2)

Unidad de aprendizaje: Sistemas Operativos

Grupo: 2CM8

*Alumnos(a):*

Briones Tapia Mariana  
Méndez Mejía Sergio Ernesto  
Nicolás Sayago Abigail  
Ramos Diaz Enrique

*Profesor(a):*

Cortes Galicia Jorge

29 de septiembre 2018

# Índice

<b>1</b>	<b>Competencias</b>	<b>2</b>
<b>2</b>	<b>Desarrollo</b>	<b>2</b>
2.1	Puntos a observar y reportar	2
2.1.1	Sección Linux:	2
2.1.2	Sección Windows:	4
2.2	Códigos fuente de los programas desarrollados	5
2.2.1	Sección Linux	5
2.2.2	Sección Windows	17
2.3	Pantallas de ejecución de los programas desarrollados	30
2.3.1	Sección Linux:	30
2.3.2	Sección Windows:	37
<b>3</b>	<b>Observaciones</b>	<b>42</b>
<b>4</b>	<b>Análisis Crítico</b>	<b>43</b>
<b>5</b>	<b>Conclusiones</b>	<b>44</b>

## 1. Competencias

El alumno aprende a familiarizarse con el administrador de procesos del sistema operativo Linux y Windows a través de la creación de procesos ligeros (hilos) para el desarrollo de aplicaciones concurrentes sencillas.

- Revisión de la creación de procesos ligeros (hilos) en Linux y Windows.
- Revisión de las llamadas al sistema y funciones de biblioteca para Hilos en ambos sistemas operativos.
- Desarrollo de aplicaciones concurrentes usando hilos tanto en Linux como en Windows.

## 2. Desarrollo

### 2.1. Puntos a observar y reportar

#### 2.1.1. Sección Linux:

Investigación de los siguientes comandos:

- ✓ **pthread\_create()** : La función `pthread_create()` inicia un nuevo hilo en la llamada proceso. El nuevo hilo comienza la ejecución invocando `start_routine()`; `arg` se pasa como el único argumento de `start_routine()`.  
El argumento `attr` apunta a una estructura `pthread_attr_t` cuyo contenido se utilizan en el momento de la creación de hilos para determinar los atributos para el nuevo hilo; esta estructura se inicializa utilizando `pthread_attr_init()`. Si `attr` es `NULL`, entonces el hilo se crea con atributos por defecto.  
Antes de regresar, una llamada exitosa a `pthread_create()` almacena la ID del nuevo hilo en el búfer apuntado por `hilo`; este identificador se utiliza para referirse al hilo en llamadas subsiguientes a otras funciones pthreads.  
En caso de éxito, `pthread_create()` devuelve 0; en caso de error, devuelve un número de error, y los contenidos de \* hilo están indefinidos.
- ✓ **pthread\_join()** : La función `pthread_join()` espera el hilo especificado por un subproceso para terminar. Si ese hilo ya ha terminado, entonces `pthread_join()` devuelve inmediatamente el hilo especificado por hilo debe poder unirse.  
Si `retval` no es `NULL`, `pthread_join()` copia el estado de salida de el subproceso de destino (es decir, el valor que el subproceso de destino suministró a `pthread_exit()` ) en la ubicación señalada por `retval` .  
En caso de éxito, `pthread_join()` devuelve 0; en caso de error, devuelve un número de error.

- ✓ **pthread\_self()** : La función `pthread_self()` devuelve el ID del hilo que llama. Este es el mismo valor que se devuelve en el hilo \* en el `pthread_create()` llamada que creó este hilo. Esta función siempre tiene éxito, devolviendo el ID del hilo que llama.
- ✓ **pthread\_exit()** : La función `pthread_exit()` termina llamada del hilo y devuelve un valor a través de `retval` que (si el hilo se puede unir) está disponible para otro hilo en el mismo proceso que llama a `pthread_join()`. No tiene retorno.
- ✓ **scandir()** : La función `scandir()` escanea el directorio `dirp`, llamando a `filter()` en cada entrada de directorio. Las entradas para las que `filter()` devuelve un valor distinto de cero son almacenados en cadenas asignadas a través de `malloc()`, ordenados usando `qsort(3)` con la función de comparación `compar()` y se recopila en la lista de nombres de matriz que se asigna a través de `malloc()`. Si el filtro es `NULL`, todas las entradas son seleccionadas.

La función `scandir()` devuelve el número de entradas de directorio seleccionadas o -1 si se produce un error.

Las funciones `alphasort()` y `versionsort()` devuelven un número entero menor que, igual o mayor que cero si se considera que el primer argumento es respectivamente menor que, igual o mayor que el segundo

- ✓ **stat()** : La llamada al sistema `stat()` devuelve información sobre un archivo. No se requieren permisos del archivo mismo, pero en el caso de `stat()` y `lstat()` - permisos de ejecución (de búsqueda) son requeridos en todos los directorios de la ruta que lleva al archivo.
  - `stat()` recupera información sobre el archivo apuntado por la ruta y esta se guarda en un buffer.
  - `lstat()` es idéntico a `stat()`, excepto que si la ruta del archivo es un enlace simbólico, entonces devuelve información sobre el enlace en sí, no del archivo al que se refiere
  - `fstat()` es idéntico a `stat()`, excepto que el archivo sobre el cual se va a recuperar información está especificado por su descriptor.

## ✓ Punto 2: Creación de un nuevo hilo

```

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
12     return 0;
13 }
14
15 void *hilo(void *arg)
16 {
17     printf("Hola Mundo desde un hilo en UNIX\n");
18     return NULL;
19 }
```

### ✓ Punto 3: Creación de hilos

```

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

```

## 2.1.2. Sección Windows:

### ✓ Punto 4: Creación de hilos

```

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

```

```

36     printf("Valores al entrar al Hilo: %i %i\n", datos->val_1, datos->val_2);
37     datos->val_1 *= 2;
38     datos->val_2 *= 2;
39     return 0;
40 }

```

## 2.2. Códigos fuente de los programas desarrollados

### 2.2.1. Sección Linux

#### ✓ Programa 5.- Creación de hilos con procesos

```

1  #include <pthread.h>
2  #include <stdio.h>
3  #include <unistd.h>
4  #include <stdlib.h>
5  #include <time.h>
6
7  void *hilo10(void *arg);
8  void *hilo5(void *arg);
9  void *hiloTerminal(void *arg);
10
11 int main(void)
12 {
13     int id_proc;
14     id_proc = fork();
15     if(id_proc == 0)
16     {
17         printf("Soy el proceso hilador\n");
18         pthread_t id_hilo;
19         for(int i = 0; i < 15; i++)
20         {
21             printf("\nHe creado %d / 15 hilos", i + 1);
22             pthread_create(&id_hilo, NULL, (void*)hilo10, NULL);
23             sleep(0);
24             for(int ii = 0; ii < 15; ii++)
25             {
26                 pthread_join(id_hilo, NULL);
27             }
28         }
29         return 0;
30     }
31     else
32     {
33         printf("Soy padre del proceso hilador\n");
34         exit(0);
35     }
36 }
37
38 void *hilo10(void *arg)
39 {
40     printf(" id = %lu", pthread_self());
41     pthread_t id_hilo;
42     for(int j = 0; j < 10; j++)
43     {
44         printf("\n\tHe creado %d / 10 hilos", j + 1);
45         pthread_create(&id_hilo, NULL, (void*)hilo5, NULL);
46         for(int jj = 0; jj < 10; jj++)
47         {
48             pthread_join(id_hilo, NULL);
49         }
50     }
51     return NULL;
52 }
53

```

```

54 void *hilo5(void *arg)
55 {
56     sleep(0);
57     printf(" id = %lu", pthread_self());
58     pthread_t id_hilo;
59     for(int i = 0; i < 5; i++)
60     {
61         printf("\n\t\tHe creado %d / 5 hilos", i + 1);
62         pthread_create(&id_hilo, NULL, (void*)hiloTerminal, NULL);
63         for(int ii = 0; ii < 5; ii++)
64         {
65             pthread_join(id_hilo, NULL);
66         }
67     }
68     return NULL;
69 }
70
71 void *hiloTerminal(void *arg)
72 {
73     printf(" id = %lu\n", pthread_self());
74     printf("\t\t\tHijo Terminal Practica 5");
75     return NULL;
76 }

```

## ✓ Programa 6.- Operaciones con matrices utilizando creación de hilos

### Creación de hilos

```

1  //   Compilación:
2  //   gcc tiempo.c -c
3  //   gcc 6.c tiempo.o -lpthread -o 6
4
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <time.h>
8  #include <stdbool.h>
9  #include <math.h>
10 #include <sys/wait.h>
11 #include <sys/types.h>
12 #include <sys/stat.h>
13 #include <fcntl.h>
14 #include <errno.h>
15 #include <unistd.h>
16 #include <string.h>
17 #include "tiempo.h"
18 #include <pthread.h>
19 #include "funciones.h"
20 #define n 10
21
22 void *hilo(void *arg);
23 char* leerDirectorio();
24
25 // CREAR DIRECTORIO
26 // Obtenemos el directorio desde la entrada de teclado
27 char* path;
28 double **matriz1, **matriz2, **suma, **resta, **mul, **tran1, **tran2, **inv1, **inv2;
29
30 int main(int argc, char const *argv[])
31 {
32     //Variables para medición de tiempos
33     double utime0, stime0, wtime0, utime1, stime1, wtime1;
34     uswtime(&utime0, &stime0, &wtime0);
35
36     int i;
37     time_t t;
38     srand((unsigned) time(&t));
39     //n = 10; // Tam de la matriz cuadrada
40     path = leerDirectorio();

```

```

41 // Inicializa las matrices.
42 matriz1 = (double**)calloc(n, sizeof(double*));
43 for (i = 0; i < n; i++)
44     matriz1[i] = (double*)calloc(n, sizeof(double));
45
46 matriz2 = (double**)calloc(n, sizeof(double*));
47 for (i = 0; i < n; i++)
48     matriz2[i] = (double*)calloc(n, sizeof(double));
49
50 suma = (double**)calloc(n, sizeof(double*));
51 for (i = 0; i < n; i++)
52     suma[i] = (double*)calloc(n, sizeof(double));
53
54 resta = (double**)calloc(n, sizeof(double*));
55 for (i = 0; i < n; i++)
56     resta[i] = (double*)calloc(n, sizeof(double));
57
58 mul = (double**)calloc(n, sizeof(double*));
59 for (i = 0; i < n; i++)
60     mul[i] = (double*)calloc(n, sizeof(double));
61
62 tran1 = (double**)calloc(n, sizeof(double*));
63 for (i = 0; i < n; i++)
64     tran1[i] = (double*)calloc(n, sizeof(double));
65
66 tran2 = (double**)calloc(n, sizeof(double*));
67 for (i = 0; i < n; i++)
68     tran2[i] = (double*)calloc(n, sizeof(double));
69
70 inv1 = (double**)calloc(n, sizeof(double*));
71 for (i = 0; i < n; i++)
72     inv1[i] = (double*)calloc(n, sizeof(double));
73
74 inv2 = (double**)calloc(n, sizeof(double*));
75 for (i = 0; i < n; i++)
76     inv2[i] = (double*)calloc(n, sizeof(double));
77
78
79 // Llamda al sistema mkdir recibe la ruta del directorio a crear,
80 // y los permisos de escritura, lectura y ejecucion para cada tipo de usuario
81 // Retorna -1 si ocurrieron errores
82 if(mkdir(path, S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH) == -1)
83 {
84     perror(path);
85     exit(EXIT_FAILURE);
86 }
87 else
88 {
89     pthread_t id_hilos[7];
90
91     // Llena matriz 1 y matriz 2
92     llenar(matriz1, n); llenar(matriz2, n);
93
94     printf("MATRIZ 1\n"); imprimir(matriz1, n);
95     printf("MATRIZ 2\n"); imprimir(matriz2, n);
96
97     for(i = 0; i<5; i++)
98     {
99         int *opcion = malloc(sizeof(int));
100         *opcion = i;
101         pthread_create (&id_hilos[i], NULL, hilo, (void*)opcion);
102     }
103     for(i = 0; i<5; i++)
104     {
105         pthread_join (id_hilos[i], NULL);
106     }
107
108     int *opc = malloc(sizeof(int));
109     *opc = 5;

```



```

110     pthread_create (&id_hilos[5], NULL, hilo, (void*)opc);
111     pthread_join (id_hilos[5], NULL);
112
113 }
114     uswtime(&utime1, &stime1, &wtime1);
115
116     //Cálculo del tiempo de ejecución del programa
117     printf("\n\nTiempo ejecucion: %.4f s\n", wtime1 - wtime0);
118
119     return 0;
120 }
121 void *hilo(void *arg)
122 {
123     int i = *((int*)arg);
124     if(i == 0)
125     {
126         // HILO SUMA
127         printf("----- Soy el hilo calculando la SUMA\n");
128         sumar(matriz1, matriz2, suma, n);
129         crearArchivo(suma, n, "/suma.txt", path);
130     }
131     if(i == 1)
132     {
133         // HILO RESTA
134         printf("----- Soy el hilo calculando la RESTA\n");
135         restar(matriz1, matriz2, resta, n);
136         crearArchivo(resta, n, "/resta.txt", path);
137     }
138     if(i == 2)
139     {
140         // HILO MULTIPLICACION
141         printf("----- Soy el hilo calculando la MULTIPLICACION\n");
142         multiplicar(matriz1, matriz2, mul, n);
143         crearArchivo(mul, n, "/mul.txt", path);
144     }
145     if(i == 3)
146     {
147         // HILO TRANSPUESTA
148         printf("----- Soy el hilo calculando la TRANSPUESTA\n");
149         transpuesta(matriz1, tran1, n);
150         crearArchivo(tran1, n, "/tran1.txt", path);
151
152         transpuesta(matriz2, tran2, n);
153         crearArchivo(tran2, n, "/tran2.txt", path);
154     }
155     if(i == 4)
156     {
157         // HILO INVERSA
158         printf("----- Soy el hilo calculando la INVERSA\n");
159
160         //Revisamos si la maztriz tiene inversa
161         if(inversa(matriz1, inv1, n) != 0)
162             crearArchivo(inv1, n, "/inversa_1.txt", path);
163
164         if(inversa(matriz2, inv2, n) != 0)
165             crearArchivo(inv2, n, "/inversa_2.txt", path);
166     }
167     if(i == 5)
168     {
169         printf(" ----- \n");
170         printf(" -- Soy el hilo leeyendo RESULTADOS -- \n");
171         printf(" ----- \n");
172
173         printf("SUMA\n"); imprimirArchivo(path, "/suma.txt");
174         printf("\nRESTA\n"); imprimirArchivo(path, "/resta.txt");
175         printf("\nMULTIPLICAR\n"); imprimirArchivo(path, "/mul.txt");
176         printf("\nTRANSPUESTA MATRIZ 1\n"); imprimirArchivo(path, "/tran1.txt");
177         printf("\nTRANSPUESTA MATRIZ 2\n"); imprimirArchivo(path, "/tran2.txt");
178         printf("\nINVERSA MATRIZ 1\n"); imprimirArchivo(path, "/inversa_1.txt");

```

```

179     printf("\nINVERSA MATRIZ 2\n"); imprimirArchivo(path, "/inversa_2.txt");
180 }
181 }
182
183
184 char* leerDirectorio()
185 {
186     char* directorio = (char*)calloc(2000, sizeof(char));
187     printf("Ingrese el nuevo directorio: ");
188     scanf("%s", directorio);
189     return directorio;
190 }

```

## Funciones con las operaciones de matrices

```

1  //  Compilación:
2  //  gcc tiempo.c -c
3  //  gcc 5.c tiempo.o -o 5
4
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <time.h>
8  #include <stdbool.h>
9  #include <math.h>
10 #include <sys/wait.h>
11 #include <sys/types.h>
12 #include <sys/stat.h>
13 #include <fcntl.h>
14 #include <errno.h>
15 #include <unistd.h>
16 #include <string.h>
17 #include "tiempo.h"
18
19 // Declaracion de funciones
20 void imprimir(double **m, int n);
21 void llenar(double **m, int n);
22 void sumar(double **m1, double **m2, double **resultado, int n);
23 void restar(double **m1, double **m2, double **resultado, int n);
24 void multiplicar(double **m1, double **m2, double **resultado, int n);
25 void transpuesta(double **m, double **resultado, int n);
26 int inversa(double **matriz, double **resultado, int n);
27 double determinante(double **matriz, int n);
28 void crearArchivo(double **matriz, int n, char *nombre, char *directorio);
29 int potencia(int base, int pot);
30 void imprimirArchivo(char *directorio, char *nombre);
31
32
33 void crearArchivo(double **matriz, int n, char *nombre, char *directorio)
34 {
35     int i, j;
36     char* dir = (char *)calloc(2000, sizeof(char));
37     char* aux = (char *)calloc(2000, sizeof(char));
38     char num[15];
39     strcpy(aux, directorio);
40     strcat(directorio, nombre);
41     strcpy(dir, directorio);
42
43     // LLamada al sistema cret, recibe la ruta del archivo a crear y los permisos
44     // Retorna -1 si hay errores
45     if(creat(directorio, S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH) == -1)
46     {
47         perror(directorio);
48         exit(EXIT_FAILURE);
49     }
50     else
51     {
52         int a = open(directorio, O_WRONLY | O_APPEND);
53         if(a == -1)

```

```

54     {
55         perror(directorio);
56         exit(EXIT_FAILURE);
57     }
58     else
59     {
60         for (i = 0; i < n; i++)
61         {
62             for(j = 0; j< n; j++)
63             {
64                 sprintf(num, "%.3f\t", matriz[i][j]);
65                 if(!write(a, num, strlen(num)) == strlen (num))
66                 {
67                     perror(directorio);
68                     exit(EXIT_FAILURE);
69                 }
70             }
71             if(!write(a, "\n", strlen("\n")) == strlen ("\n"))
72             {
73                 perror(directorio);
74                 exit(EXIT_FAILURE);
75             }
76         }
77     }
78     close(a);
79 }
80 strcpy(directorio, aux);
81 free(aux); free(dir);
82 }
83
84 void imprimirArchivo(char *directorio, char *nombre)
85 {
86     char* dir = (char *)calloc(2000, sizeof(char));
87     char* aux = (char *)calloc(2000, sizeof(char));
88     strcpy(aux, directorio);
89     strcpy(dir, directorio);
90     strcat(dir, nombre);
91
92     int archivo = open(dir, O_RDONLY);
93     if(archivo == -1)
94     {
95         perror(dir);
96         exit(EXIT_FAILURE);
97     }
98     struct stat sb;
99     if(stat(dir, &sb) == -1)
100     {
101         perror(dir);
102         exit(EXIT_FAILURE);
103     }
104     long long longitud = (long long) sb.st_size;
105     char *contenido = (char *)calloc(longitud, sizeof(char));
106
107     if(read(archivo, contenido, longitud) == longitud)
108     {
109         printf("%s", contenido);
110     }
111     if(close(archivo) == -1)
112     {
113         perror(dir);
114         exit(EXIT_FAILURE);
115     }
116     free(contenido);
117     strcpy(directorio, aux);
118 }
119
120
121
122 void imprimir(double **m, int n)

```

```

123 {
124     int i, j;
125     for(i = 0; i < n; i++)
126     {
127         for(j = 0; j < n; j++)
128             printf("%.3f\t", m[i][j]);
129         printf("\n");
130     }
131     printf("\n");
132 }
133
134 // Llena con numeros random
135 void llenar(double **m, int n)
136 {
137     int i, j;
138     for (i = 0; i < n; i++)
139     {
140         for (j = 0; j < n; j++)
141         {
142             m[i][j] = (rand()%11);
143         }
144     }
145 }
146
147 void sumar(double **m1, double **m2, double **resultado, int n)
148 {
149     int i, j;
150     for(i = 0; i < n; i++)
151     {
152         for(j = 0; j < n; j++)
153             resultado[i][j] = m1[i][j] + m2[i][j];
154     }
155 }
156
157 void restar(double **m1, double **m2, double **resultado, int n)
158 {
159     int i, j;
160     for(i = 0; i < n; i++)
161     {
162         for(j = 0; j < n; j++)
163             resultado[i][j] = m1[i][j] - m2[i][j];
164     }
165 }
166
167 void multiplicar(double **m1, double **m2, double **resultado, int n)
168 {
169     int i, j, k, aux;
170     for(i = 0; i < n; i++)
171     {
172         for(j = 0; j < n; j++)
173         {
174             aux = 0;
175             for(k = 0; k < n; k++)
176                 aux = m1[i][k] * m2[k][j] + aux;
177             resultado[i][j] = aux;
178         }
179     }
180 }
181
182 void transpuesta(double **m, double **resultado, int n)
183 {
184     int i, j;
185     for(i = 0; i < n; i++)
186     {
187         for(j = 0; j < n; j++)
188             resultado[i][j] = m[j][i];
189     }
190 }
191

```

```

192 bool esCero(double x)
193 {
194     return fabs(x) < 1e-8;
195 }
196
197 double determinante(double **m, int n)
198 {
199     double det = 0, aux = 0;
200     int c;
201     // Si el orden es de 2, multiplica cruzadon directamente
202     if(n==2)
203         return m[0][0]*m[1][1] - m[1][0]*m[0][1];
204     else
205     {
206         for(int j=0; j<n; j++)
207         {
208             // Crea arreglo dinamico temporal
209             double **menor = (double **)malloc(sizeof(double)*(n-1));
210             // Redimensiona
211             for(int i=0; i<(n-1); i++)
212                 menor[i] = (double *)malloc(sizeof(double)*(n-1));
213             for(int k=1; k<n; k++)
214             {
215                 c = 0;
216                 for(int l=0; l<n; l++)
217                 {
218                     if(l!=j)
219                     {
220                         /*Parte matriz principal en matrices de 3
221                         y multiplica cruzado*/
222                         menor[k-1][c] = m[k][l];
223                         c++;
224                     }
225                 }
226                 // Recursividad, repite la funcion
227                 aux = potencia(-1, 2+j)*m[0][j]*determinante(menor, n-1);
228                 det += aux;
229
230                 for(int x = 0; x<(n-1); x++)
231                     free(menor[x]); // Libera espacio en memoria
232                 free(menor);
233             }
234         }
235         return det; // Devuelve resultado
236     }
237 }
238
239 // Usando definicion de la adjunta
240 int inversa(double **A, double **resultado, int n)
241 {
242     int tieneInversa;
243     if(determinante(A, n) == 0)
244     {
245         tieneInversa=0;
246         printf("La matriz no tiene inversa. Determinante = 0\n\n");
247     }
248     else
249     {
250         tieneInversa=1;
251         int i, j, k, l;
252         double *tmp;
253         tmp = (double*)malloc(sizeof(double)*n);
254
255         for(i = 0; i < n; ++i)
256             resultado[i][i] = 1;
257         i = 0; j = 0;
258         while(i < n && j < n)
259         {
260             if(esCero(A[i][j]))

```

```

261     {
262         for(k = i + 1; k < n; ++k)
263         {
264             if(!esCero(A[k][j]))
265             {
266                 tmp = A[i];
267                 A[i] = A[k];
268                 A[k] = tmp;
269                 tmp = resultado[i];
270                 resultado[i] = resultado[k];
271                 resultado[k] = tmp;
272                 break;
273             }
274         }
275     }
276     if(!esCero(A[i][j]))
277     {
278         for(l = 0; l < n; ++l)
279             resultado[i][l] /= A[i][j];
280         for(l = n - 1; l >= j; --l)
281             A[i][l] /= A[i][j];
282         for(k = 0; k < n; ++k)
283         {
284             if(i == k) continue;
285             for(l = 0; l < n; ++l)
286                 resultado[k][l] -= resultado[i][l] * A[k][j];
287             for(l = n; l >= j; --l)
288                 A[k][l] -= A[i][l] * A[k][j];
289         }
290         ++i;
291     }
292     ++j;
293 }
294 }
295 return tieneInversa;
296 }
297
298 int potencia(int base, int pot)
299 {
300     int i, resultado = 1;
301     for(i = 0; i < pot; i++)
302         resultado = base * resultado;
303
304     return resultado;
305 }

```

### ✓ Programa 7.- Creación de directorios y copia de archivos concurrente utilizando creación de hilos

```

1 //Compilar: gcc 7.c -lpthread -o 7
2 //Ejecutar: ./7 [RutaOrigen] [RutaDestino]
3
4 #include <stdio.h>
5 #include <string.h>
6 #include <stdlib.h>
7 #include <sys/types.h>
8 #include <sys/stat.h>
9 #include <unistd.h>
10 #include <fcntl.h>
11 #include <dirent.h>
12 #include <pthread.h>
13
14 //Estructura para manejar la ruta origen y destino
15 typedef struct{
16     char Origen[400];
17     char Destino[400];
18 } RUTA;
19

```

```

20 //Función que ignora los archivos y solo lee directorios
21 int esDirectorio(char path[])
22 {
23     //Nos ayuda para consultar info del directorio/archivo escaneado
24     struct stat sb;
25
26     if(stat(path, &sb) == -1)
27     {
28         perror(path);
29         exit(-1);
30     }
31
32     //Revisamos el tipo
33     if(S_ISDIR(sb.st_mode))
34         return 1;
35     else
36         return 0;
37 }
38
39 void concatenar(char dir[], char aux[], char nombre[])
40 {
41     //Concatener directorio y archivo
42     strcpy(dir, aux);
43     strcat(dir, "/");
44     strcat(dir, nombre);
45 }
46
47 int ignorarPuntos(char nombre[])
48 {
49     //Al escanear un directorio, aparecen puntos
50     //Con esta función, los eliminamos para trabajar solo con directorios/archivos
51     if(strncmp(nombre, ".") && strcmp(nombre, ".."))
52         return 1;
53     else
54         return 0;
55 }
56
57 void copiarArchivo(char origen[], char destino[])
58 {
59     ssize_t leer, escribir;
60     //Llamada al sistema open recibe la ruta del archivo a abrir y el modo (lectura, escritura,
61     //↪ ejecución)
62     //Devuelve un descriptor de archivo
63     int o = open(origen, O_RDONLY); //Abrimos la ruta origen
64
65     //Llamada al sistema creat recibe la ruta del archivo a crear y los permisos
66     //Retorna -1 si existieron errores
67     if(creat(destino, S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH) == -1)
68     {
69         perror(destino);
70         exit(EXIT_FAILURE);
71     }
72
73     //Abrimos la ruta destino
74     int d = open(destino, O_WRONLY | O_APPEND);
75
76     //Nos ayuda para consultar info del directorio/archivo escaneado
77     struct stat sorigen;
78     if(stat(origen, &sorigen) == -1)
79     {
80         perror(origen);
81         exit(EXIT_FAILURE);
82     }
83
84     //Obtenemos la longitud en bytes del archivo con ayuda de stat
85     long long longitud = (long long) sorigen.st_size;
86     //Creamos un buffer para ir guardando el contenido del archivo
87     char *contenido = (char *)calloc(longitud, sizeof(char));

```

```

88     if(o == -1)
89     {
90         perror(origen);
91         exit(-1);
92     }
93
94     if(d == -1)
95     {
96         perror(destino);
97         exit(-1);
98     }
99
100    //Leemos el archivo origen y lo escribimos en el destino
101    while((leer = read(o, contenido, longitud)) == longitud)
102    {
103        escribir = write(d, contenido, longitud);
104        //Establecemos el fin del contenido
105        memset(contenido, '\0', longitud);
106        printf("%s ..... COPIADO", destino);
107        printf("\n\n");
108    }
109
110    close(o);
111    close(d);
112 }
113
114 //Funciones que ejecutaran los hilos creados
115 void *hilo(void *arg)
116 {
117     //Utilizamos la estructura que creamos
118     RUTA *directorios, *raices = (RUTA*)arg;
119     //Nos sirve para leer todos los directorios contenidos en una ruta
120     struct dirent **dirp;
121     //Creamos un hilo
122     pthread_t *id_hilo = (pthread_t*)calloc(1, sizeof(pthread_t));
123     int num, i, t = 0;
124
125     //Leemos la ruta origen en busca de directorios
126     num = scandir(raices->Origen, &dirp, NULL, alphasort);
127
128     if(num == -1)
129     {
130         perror(raices->Origen);
131         exit(-1);
132     }
133     else if (num > 0)
134     {
135
136         //Reasignamos memoria para guardar los directorios encontrados
137         directorios = (RUTA*)calloc(num, sizeof(RUTA));
138         for(i = 0; num-- > 0; i++)
139         {
140             //Vamos construyendo las rutas de los directorios origen
141             concatenar(directorios[i].Origen, raices->Origen, dirp[num]->d_name);
142             if(esDirectorio(directorios[i].Origen))
143             {
144                 if(ignorarPuntos(dirp[num]->d_name))
145                 {
146                     printf("Soy el hilo %d. Encontre: %s", i, directorios[i].Origen);
147                     printf("\n");
148
149                     //Solo nos quedamos con directorios y construimos las rutas de destino
150                     concatenar(directorios[i].Destino, raices->Destino, dirp[num]->d_name);
151                     printf("Soy el hilo %d. Creare: %s", i, directorios[i].Destino);
152                     printf("\n\n");
153                     //Creamos los directorios destino
154                     if(mkdir(directorios[i].Destino, S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH) == -1)
155                     {
156                         perror(directorios[i].Destino);

```



```

157         exit(-1);
158     }
159     //Vamos creando un nuevo hilo por cada directorio encontrado
160     id_hilo = (pthread_t*)realloc(id_hilo, sizeof(pthread_t)*(t+1));
161     pthread_create(&id_hilo[t], NULL, hilo, (void*)&directorios[i]);
162     t++;
163 }
164 }
165 else
166 {
167     //Ahora, creamos las rutas para los archivos dentro de cada directorio
168     concatenar(directorios[i].Destino, raices->Destino, dirp[num]->d_name);
169     printf("Copiare el archivo: %s", directorios[i].Origen);
170     printf("\n");
171     //Leemos y escribimos los archivos
172     copiarArchivo(directorios[i].Origen, directorios[i].Destino);
173 }
174 }
175 //Esperamos a que todos los hilos acaben
176 for(i = 0; i < t; i++)
177     pthread_join(id_hilo[i], NULL);
178
179 free(id_hilo);
180 free(directorios);
181 }
182
183 }
184
185 int main(int argc, char *argv[])
186 {
187     //Creamos las rutas origen y destino
188     RUTA *rutas = (RUTA*)calloc(1, sizeof(RUTA));
189     pthread_t id_hilo;
190
191     //Si no se ingresaron rutas por linea de comandos, tomamos las predeterminadas
192     if(argc != 3)
193     {
194         strcpy(rutas->Origen, "/home/enrike/Escritorio/origen");
195         strcpy(rutas->Destino, "/home/enrike/Escritorio/destino");
196     }
197     else
198     {
199         strcpy(rutas->Origen, argv[1]);
200         strcpy(rutas->Destino, argv[2]);
201     }
202
203     printf("Ruta Origen: %s", rutas->Origen);
204     printf("\n");
205     printf("Ruta Destino: %s", rutas->Destino);
206     printf("\n\n\n");
207     /*Revisamos la ruta ingresada es un directorio, y que los directorios de origen
208     no existan en la ruta destino*/
209     if(esDirectorio(rutas->Origen))
210     {
211         if(mkdir(rutas->Destino, S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH) == -1)
212         {
213             perror(rutas->Destino);
214             exit(-1);
215         }
216         else
217         {
218             //Creamos un hilo y enviamos las rutas
219             pthread_create(&id_hilo, NULL, hilo, (void*)rutas);
220             pthread_join(id_hilo, NULL);
221         }
222     }
223     return 0;
224 }

```

### 2.2.2. Sección Windows

#### ✓ Programa 5.- Creación de hilos con procesos

##### Proceso padre

```

1  #include <windows.h>
2  #include <stdio.h>
3
4  int main(int argc, char const *argv[])
5  {
6      STARTUPINFO si;           //Estructura de informacion inicial para windows
7      PROCESS_INFORMATION pi;   //Estructura de informacion del admn. de procesos
8      int i;
9      ZeroMemory(&si, sizeof(si));
10     si.cb = sizeof(si);
11     ZeroMemory(&pi, sizeof(pi));
12     if (argc != 2)
13     {
14         return;
15     }
16     //Creacion proceso hijo
17     if (!CreateProcess(NULL, argv[1], NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
18     {
19         printf("Fallo al invocar CreateProcess (%d)\n", GetLastError() );
20         return;
21     }
22     //Proceso padre
23     printf("Soy el padre del proceso hilador\n");
24     WaitForSingleObject(pi.hProcess, INFINITE);
25     //Terminacion controlada del proceso e hilo asociado de ejecucion
26     CloseHandle(pi.hProcess);
27     CloseHandle(pi.hThread);
28 }

```

##### Proceso hijo - Creación de hilos

```

1  #include <windows.h>
2  #include <stdio.h>
3  DWORD WINAPI Hilos10(LPVOID lpParam);
4  DWORD WINAPI Hilos5(LPVOID lpParam);
5  DWORD WINAPI HiloTerminal(LPVOID lpParam);
6  typedef struct informacion info;
7  struct informacion
8  {
9      int val1;
10 };
11 int i, j, k, l;
12
13 int main(void)
14 {
15     DWORD idHilo;
16     HANDLE manHilo;
17     info argumentos;
18     printf("\n Soy el proceso hilador");
19     for(i = 0; i < 15; i++)
20     {
21         printf("\nHe creado %d de 15 hilos", i+1);
22         manHilo = CreateThread(NULL, 0, Hilos10, &argumentos, 0, &idHilo);
23         for(l = 0; l < 15; l++)
24         {
25             WaitForSingleObject(manHilo, INFINITE);
26         }
27     }
28
29     CloseHandle(manHilo);
30     return 0;

```

```

31 }
32
33 DWORD WINAPI Hilos10(LPVOID lpParam)
34 {
35     printf(": id = %d", GetCurrentThreadId());
36     DWORD idHilo;
37     HANDLE manHilo;
38     info argumentos;
39     for(k = 0; k < 10; k++)
40     {
41         printf("\n\tHe creado %d de 10 hilos",k+1);
42         manHilo = CreateThread(NULL, 0, Hilos5,&argumentos,0,&idHilo);
43         WaitForSingleObject(manHilo, INFINITE);
44     }
45     CloseHandle(manHilo);
46     return 0;
47 }
48
49 DWORD WINAPI Hilos5(LPVOID lpParam)
50 {
51     printf(": id = %d", GetCurrentThreadId());
52     DWORD idHilo;
53     HANDLE manHilo;
54     info argumentos;
55     for(j = 0; j < 5; j++)
56     {
57         printf("\n\t\tHe creado %d de 5 hilos",j+1);
58         manHilo = CreateThread(NULL, 0, HiloTerminal,&argumentos,0,&idHilo);
59         WaitForSingleObject(manHilo, INFINITE);
60     }
61     CloseHandle(manHilo);
62     return 0;
63 }
64
65 DWORD WINAPI HiloTerminal(LPVOID lpParam)
66 {
67     printf(": id = %d", GetCurrentThreadId());
68     printf("\n\t\t\tPractica 5, Este es un Hilo terminal %d", GetCurrentThreadId());
69     return 0;
70 }

```

## ✓ Programa 6.- Operaciones con matrices utilizando creación de hilos

### Creación de hilos

```

1 // Compilación: gcc 6.c -o 6
2 // Ejecutar: 6 [RutaDestino]
3
4 #include <windows.h>
5 #include <time.h>
6 #include <errno.h>
7 #include <stdio.h>
8 #include <stdlib.h>
9 #include <string.h>
10 #include <stdbool.h>
11 #include <math.h>
12 #include "funciones.h"
13 #define n 10
14
15 double **matriz1, **matriz2, **suma, **resta, **mul, **tran1, **tran2, **inv1, **inv2;
16 char* path;
17
18 DWORD WINAPI funcionHilo(LPVOID lpParam);
19
20 int main(int argc, char const *argv[])
21 {
22     // Para medir el tiempo
23     clock_t tiempo_inicio, tiempo_final;

```

```

24     double segundos;
25
26     tiempo_inicio = clock();
27     int i;
28     time_t t;
29     srand((unsigned) time(&t));
30
31     // Inicializa las matrices.
32     matriz1 = (double**)calloc(n, sizeof(double*));
33     for (i = 0; i < n; i++)
34         matriz1[i] = (double*)calloc(n, sizeof(double));
35
36     matriz2 = (double**)calloc(n, sizeof(double*));
37     for (i = 0; i < n; i++)
38         matriz2[i] = (double*)calloc(n, sizeof(double));
39     resta = (double**)calloc(n, sizeof(double*));
40     for (i = 0; i < n; i++)
41         resta[i] = (double*)calloc(n, sizeof(double));
42
43     mul = (double**)calloc(n, sizeof(double*));
44     for (i = 0; i < n; i++)
45         mul[i] = (double*)calloc(n, sizeof(double));
46
47     tran1 = (double**)calloc(n, sizeof(double*));
48     for (i = 0; i < n; i++)
49         tran1[i] = (double*)calloc(n, sizeof(double));
50
51     tran2 = (double**)calloc(n, sizeof(double*));
52     for (i = 0; i < n; i++)
53         tran2[i] = (double*)calloc(n, sizeof(double));
54
55     inv1 = (double**)calloc(n, sizeof(double*));
56     for (i = 0; i < n; i++)
57         inv1[i] = (double*)calloc(n, sizeof(double));
58
59     inv2 = (double**)calloc(n, sizeof(double*));
60     for (i = 0; i < n; i++)
61         inv2[i] = (double*)calloc(n, sizeof(double));
62
63     if(argc != 2)
64         exit(-1);
65
66     path = (char*)calloc(2000, sizeof(char));
67     strcpy(path, argv[1]);
68
69     if(!CreateDirectory(path, NULL))
70     {
71         perror(path);
72         exit(-1);
73     }
74
75     // Llena matriz 1 y matriz 2
76     llenar(matriz1, n);
77     llenar(matriz2, n);
78
79     printf("MATRIZ 1\n"); imprimir(matriz1, n);
80     printf("MATRIZ 2\n"); imprimir(matriz2, n);
81
82     // ----- HILOS -----
83     HANDLE manHilo[7]; // Manejador del Hilo
84     DWORD idHilo[7]; // Identificador de los hilos
85
86     for(i = 0; i < 5; i++)
87     {
88         // Lleva el control del hilo a realizar
89         int *opcion = malloc(sizeof(int));
90         *opcion = i;
91         // Creacion del hilo [i]
92         manHilo[i] = CreateThread(NULL, 0, funcionHilo, opcion, 0, &idHilo[i]);

```

```

93     }
94
95     // Espera la finalizacion del hilo
96     for(i = 0; i < 5; i++)
97     {
98         WaitForSingleObject(manHilo[i], INFINITE);
99         // Cierre del manejador del hilo creado
100        CloseHandle(manHilo[i]);
101    }
102
103    // Ultimo HILO - Leer archivos
104    int *opc = malloc(sizeof(int));
105    *opc = 5;
106    manHilo[5] = CreateThread(NULL, 0, funcionHilo, opc, 0, &idHilo[5]);
107    WaitForSingleObject(manHilo[5], INFINITE);
108    // Cierre del manejador del hilo creado
109    CloseHandle(manHilo[5]);
110
111    tiempo_final = clock();
112    segundos = (double)(tiempo_final - tiempo_inicio) / CLOCKS_PER_SEC;
113    //Cálculo del tiempo de ejecución del programa
114    printf("\n\nTiempo ejecucion: %.4f s\n", segundos);
115
116    return 0;
117 }
118
119 DWORD WINAPI funcionHilo(LPVOID lpParam)
120 {
121     int opc = *(int *)lpParam;
122     int i;
123     if(opc == 0)
124     {
125         //HILO SUMA
126         suma = (double**)calloc(n, sizeof(double*));
127         for (i = 0; i < n; i++)
128             suma[i] = (double*)calloc(n, sizeof(double));
129
130         printf("Soy el hilo calculando la SUMA\n"); sumar(matriz1, matriz2, suma, n);
131         crearArchivo(suma, path, "suma.txt");
132     }
133     else if(opc == 1)
134     {
135         //HILO RESTA
136         printf("Soy el hilo calculando la RESTA\n"); restar(matriz1, matriz2, resta, n);
137         crearArchivo(resta, path, "resta.txt");
138     }
139     else if(opc == 2)
140     {
141         //HILO MULTIPLICACION
142         printf("Soy el hilo calculando la MULTIPLICAR\n"); multiplicar(matriz1, matriz2, mul,
143             ↵ n);
144         crearArchivo(mul, path, "multiplicacion.txt");
145     }
146     else if(opc == 3)
147     {
148         //HILO TRANSPUESTA
149         printf("Soy el hilo calculando la TRANSPUESTA MATRIZ 1\n"); transpuesta(matriz1, tran1,
150             ↵ n);
151         crearArchivo(tran1, path, "transpuesta_1.txt");
152
153         printf("Soy el hilo calculando la TRANSPUESTA MATRIZ 2\n"); transpuesta(matriz2, tran2,
154             ↵ n);
155         crearArchivo(tran2, path, "transpuesta_2.txt");
156     }
157     else if(opc == 4)
158     {
159         //HILO INVERSA
160         printf("Soy el hilo calculando la INVERSA MATRIZ 1\n");
161         //Revisamos si la maztriz tiene inversa

```

```

159     if(inversa(matriz1, inv1, n) != 0)
160         crearArchivo(inv1, path, "inversa_1.txt");
161
162     printf("Soy el hilo calculando la INVERSA MATRIZ 2\n");
163     if(inversa(matriz2, inv2, n) != 0)
164         crearArchivo(inv2, path, "inversa_2.txt");
165 }
166 else
167 {
168     //HILO RESULTADO
169     printf(" -----\n");
170     printf(" -- Soy el hilo leyendo RESULTADOS --\n");
171     printf(" -----\n");
172
173     printf("SUMA\n"); imprimirArchivo(path, "suma.txt");
174     printf("\nRESTA\n"); imprimirArchivo(path, "resta.txt");
175     printf("\nMULTIPLICAR\n"); imprimirArchivo(path, "multiplicacion.txt");
176     printf("\nTRANSPUESTA MATRIZ 1\n"); imprimirArchivo(path, "transpuesta_1.txt");
177     printf("\nTRANSPUESTA MATRIZ 2\n"); imprimirArchivo(path, "transpuesta_2.txt");
178     printf("\nINVERSA MATRIZ 1\n"); imprimirArchivo(path, "inversa_1.txt");
179     printf("\nINVERSA MATRIZ 2\n"); imprimirArchivo(path, "inversa_2.txt");
180 }
181 return 0;
182 }

```

## Funciones con las operaciones de matrices

```

1  #include <windows.h>
2  #include <time.h>
3  #include <errno.h>
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <stdbool.h>
8  #include <math.h>
9
10 // Declaracion de funciones
11 int potencia(int base, int pot);
12 void imprimir(double **m, int n);
13 void llenar(double **m, int n);
14 void sumar(double **m1, double **m2, double **resultado, int n);
15 void restar(double **m1, double **m2, double **resultado, int n);
16 void multiplicar(double **m1, double **m2, double **resultado, int n);
17 void transpuesta(double **m, double **resultado, int n);
18 int inversa(double **matriz, double **resultado, int n);
19 double determinante(double **matriz, int n);
20 void crearArchivo(double **res, char* dir, char *nombre);
21 void imprimirArchivo(char *directorio, char *nombre);
22
23 void crearArchivo(double **res, char* dir, char *nombre)
24 {
25     int i,j;
26     char num[15];
27     char *name = (char *)calloc(150,sizeof(char));
28     strcpy(name, nombre);
29     char *ruta = (char *)calloc(150,sizeof(char));
30     // Contatenamos la ruta original con el nombre del archivo
31     strcat(strcat(strcpy(ruta, dir), "\\"), name);
32
33     HANDLE h = CreateFile(ruta, //ruta del archivo
34                           GENERIC_WRITE, //abrir para escribir
35                           0, //no compartir
36                           NULL, // seguridad por default
37                           CREATE_ALWAYS, //crear siempre
38                           FILE_ATTRIBUTE_TEMPORARY, //archivo normal
39                           NULL); //sin tributos
40
41     if (h == INVALID_HANDLE_VALUE)

```

```

42     {
43         perror(ruta);
44         exit(EXIT_FAILURE);
45     }
46     else
47     {
48         DWORD bytesEscritos = 0;
49
50         for(i=0 ; i<10 ; i++) // Escribimos 5 veces el texto en el archivo
51         {
52             for(j=0 ; j<10 ; j++) // Escribimos 5 veces el texto en el archivo
53             {
54                 sprintf(num, "%.3f\t", res[i][j]);
55                 /*Funcion WriteFile recibe los parametros a continuacion y devuelve un true si
56                  ↳ no existieron errores*/
57                 BOOL escribir = WriteFile(
58                     h,                                     // abrir handle del archivo
59                     num,                                   // informacion a escribir
60                     (DWORD)strlen(num),                   // tamaño de bytes a escribir
61                     &bytesEscritos,                       // tamaño de bytes escrit
62                     NULL);                                // no overlapped structure
63
64                 if(!escribir)
65                 {
66                     perror(ruta);
67                     exit(EXIT_FAILURE);
68                 }
69                 BOOL espacio = WriteFile(
70                     h,                                     // abrir handle del archivo
71                     "\n",                                 // informacion a escribir
72                     (DWORD)strlen("\n"),                 // tamaño de bytes a escribir
73                     &bytesEscritos,                       // tamaño de bytes escrit
74                     NULL);                                // no overlapped structure
75
76                 if(!espacio)
77                 {
78                     perror(ruta);
79                     exit(EXIT_FAILURE);
80                 }
81                 //Llamada al sistema CloseHandle recibe un descriptor de archivo y retorna un valor
82                 ↳ cero si han habido errores
83                 if(CloseHandle(h) == 0)
84                 {
85                     perror(ruta);
86                     exit(EXIT_FAILURE);
87                 }
88             }
89         }
90     }
91
92 void imprimirArchivo(char *directorio, char *nombre)
93 {
94     char *name = (char *)calloc(150, sizeof(char));
95     strcpy(name, nombre);
96     char *dir = (char *)calloc(150, sizeof(char));
97     // Contatenamos la ruta original con el nombre del archivo
98     strcat(strcat(strcpy(dir, directorio), "\\"), name);
99
100     HANDLE file;
101     DWORD BytesEscritos = 0;
102     char *contenido = (char *)calloc(1000000, sizeof(char));
103     file = CreateFile(
104         dir,
105         GENERIC_WRITE | GENERIC_READ,
106         FILE_SHARE_READ,
107         NULL,
108         OPEN_EXISTING,
109         FILE_ATTRIBUTE_NORMAL,
110         NULL);

```

```

109
110     if(file == INVALID_HANDLE_VALUE)
111     {
112         printf("Error 1\n");
113         perror(dir);
114         exit(EXIT_FAILURE);
115     }
116     else
117     {
118         if(ReadFile(file, contenido, 1000000, &BytesEscritos, NULL))
119         {
120             printf("%s", contenido);
121         }
122         free(contenido);
123
124         if(CloseHandle(file) == 0)
125         {
126             perror(dir);
127             exit(EXIT_FAILURE);
128         }
129     }
130     free(name);
131     free(dir);
132 }
133
134 void imprimir(double **m, int n)
135 {
136     int i, j;
137     for(i = 0; i < n; i++)
138     {
139         for(j = 0; j < n; j++)
140             printf("%.3f\t", m[i][j]);
141         printf("\n");
142     }
143     printf("\n");
144 }
145
146 // Llena con numeros random
147 void llenar(double **m, int n)
148 {
149     int i, j;
150     for (i = 0; i < n; i++)
151     {
152         for (j = 0; j < n; j++)
153         {
154             m[i][j] = (rand()%11);
155         }
156     }
157 }
158
159 void sumar(double **m1, double **m2, double **resultado, int n)
160 {
161     int i, j;
162     for(i = 0; i < n; i++)
163     {
164         for(j = 0; j < n; j++)
165             resultado[i][j] = m1[i][j] + m2[i][j];
166     }
167 }
168
169 void restar(double **m1, double **m2, double **resultado, int n)
170 {
171     int i, j;
172     for(i = 0; i < n; i++)
173     {
174         for(j = 0; j < n; j++)
175             resultado[i][j] = m1[i][j] - m2[i][j];
176     }
177 }

```



```

178
179 void multiplicar(double **m1, double **m2, double **resultado, int n)
180 {
181     int i, j, k, aux;
182     for(i = 0; i < n; i++)
183     {
184         for(j = 0; j < n; j++)
185         {
186             aux = 0;
187             for(k = 0; k < n; k++)
188                 aux = m1[i][k] * m2[k][j] + aux;
189             resultado[i][j] = aux;
190         }
191     }
192 }
193
194 void transpuesta(double **m, double **resultado, int n)
195 {
196     int i, j;
197     for(i = 0; i < n; i++)
198     {
199         for(j = 0; j < n; j++)
200             resultado[i][j] = m[j][i];
201     }
202 }
203 bool esCero(double x)
204 {
205     return fabs(x) < 1e-8;
206 }
207
208 double determinante(double **m, int n)
209 {
210     double det = 0, aux = 0;
211     int c, i, j, k, l, x;
212     //Si el orden es de 2, multiplica cruzadon directamente
213     if(n==2)
214         return m[0][0]*m[1][1] - m[1][0]*m[0][1];
215     else
216     {
217         for(j=0; j<n; j++)
218         {
219             //Crea arreglo dinamico temporal
220             double **menor = (double **)malloc(sizeof(double)*(n-1));
221             //Redimensiona
222             for(i=0; i<(n-1); i++)
223                 menor[i] = (double *)malloc(sizeof(double)*(n-1));
224             for(k=1; k<n; k++)
225             {
226                 c = 0;
227                 for(l=0; l<n; l++)
228                 {
229                     if(l!=j)
230                     {
231                         /*Parte matriz principal en matrices de 3
232                         y multiplica cruzado*/
233                         menor[k-1][c] = m[k][l];
234                         c++;
235                     }
236                 }
237             }
238             //Recurividad, repite la funcion
239             aux = potencia(-1, 2+j)*m[0][j]*determinante(menor, n-1);
240             det += aux;
241         }
242         for(x = 0; x<(n-1); x++)
243             free(menor[x]); //Libera espacio en memoria
244         free(menor);
245     }
246     return det; //Devuelve resultado

```

```

247     }
248 }
249
250 // Usando definicion de la adjunta
251 int inversa(double **A, double **resultado, int n)
252 {
253     int tieneInversa;
254     if(determinante(A, n) == 0)
255     {
256         tieneInversa=0;
257         printf("La matriz no tiene inversa. Determinante = 0\n\n");
258     }
259     else{
260         tieneInversa=1;
261         int i, j, k, l;
262         double *tmp;
263         tmp = (double*)malloc(sizeof(double)*n);
264
265         for(i = 0; i < n; ++i)
266             resultado[i][i] = 1;
267         i = 0; j = 0;
268         while(i < n && j < n)
269         {
270             if(esCero(A[i][j]))
271             {
272                 for(k = i + 1; k < n; ++k)
273                 {
274                     if(!esCero(A[k][j]))
275                     {
276                         tmp = A[i];
277                         A[i] = A[k];
278                         A[k] = tmp;
279                         tmp = resultado[i];
280                         resultado[i] = resultado[k];
281                         resultado[k] = tmp;
282                         break;
283                     }
284                 }
285             }
286             if(!esCero(A[i][j]))
287             {
288                 for(l = 0; l < n; ++l)
289                     resultado[i][l] /= A[i][j];
290                 for(l = n - 1; l >= j; --l)
291                     A[i][l] /= A[i][j];
292                 for(k = 0; k < n; ++k)
293                 {
294                     if(i == k) continue;
295                     for(l = 0; l < n; ++l)
296                         resultado[k][l] -= resultado[i][l] * A[k][j];
297                     for(l = n; l >= j; --l)
298                         A[k][l] -= A[i][l] * A[k][j];
299                 }
300                 ++i;
301             }
302             ++j;
303         }
304     }
305     return tieneInversa;
306 }
307
308 int potencia(int base, int pot)
309 {
310     int i, resultado = 1;
311     for(i = 0; i < pot; i++)
312         resultado = base * resultado;
313
314     return resultado;
315 }

```

## ✓ Programa 7.- Creación de directorios y copia de archivos concurrente utilizando creación de hilos

```

1  //Compilar: gcc 7.c -o 7
2  //Ejecutar: 7 [RutaOrigen] [RutaDestino]
3
4  #include <stdio.h>
5  #include <windows.h>
6  #include <string.h>
7  #include <stdlib.h>
8
9  //Estructura para manejar la ruta origen y destino
10 typedef struct r RUTA;
11 struct r
12 {
13     char Origen[400];
14     char Destino[400];
15 };
16
17 int esDirectorio(WIN32_FIND_DATA ruta)
18 {
19     //Revisamos si lo que se encontro en el escaneo es un directorio
20     if (ruta.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY)
21         return 1;
22     else
23         return 0;
24 }
25
26 int ignorarPuntos(char nombre[])
27 {
28     //Al escanear un directorio, aparecen puntos
29     //Con este funcion, los eliminamos para trabajar solo con directorios/archivos
30     if (strcmp(nombre, ".") && strcmp(nombre, ".."))
31         return 1;
32     else
33         return 0;
34 }
35
36
37 void concatenar(char dir[], char aux[], char nombre[])
38 {
39     //Concatener directorio y archivo
40     strcpy(dir, aux);
41     strcat(dir, "\\");
42     strcat(dir, nombre);
43 }
44
45 void copiarArchivo(char origen[], char destino[])
46 {
47     //Handle para leer
48     HANDLE o = CreateFile(origen, //ruta del archivo
49                           GENERIC_READ, //abrir para leer-escribir
50                           FILE_SHARE_READ, //no compartir
51                           NULL, // seguridad por default
52                           OPEN_EXISTING, //crear siempre
53                           FILE_ATTRIBUTE_NORMAL, //archivo normal
54                           NULL);
55
56     if (o == INVALID_HANDLE_VALUE)
57     {
58         perror(origen);
59         exit(EXIT_FAILURE);
60     }
61
62     //Handle para escribir
63     HANDLE d = CreateFile(destino, //ruta del archivo
64                           GENERIC_WRITE, //abrir para escribir
65                           0, //no compartir
66                           NULL, // seguridad por default

```

```

67             CREATE_ALWAYS,           //crear siempre
68             FILE_ATTRIBUTE_NORMAL,    //archivo normal
69             NULL);
70
71     if (d == INVALID_HANDLE_VALUE)
72     {
73         perror(destino);
74         exit(EXIT_FAILURE);
75     }
76
77     DWORD bytesEscritos = 0;
78     char *contenido = (char *)calloc(1000000, sizeof(char));
79
80     //Leemos el archivo origen y lo escribimos en el destino
81     if(ReadFile(o, contenido, 1000000, &bytesEscritos, NULL))
82     {
83         BOOL escribir = WriteFile(d,           // abrir handle del archivo
84                                   contenido,    // informacion a escribir
85                                   (DWORD)strlen(contenido), // tamaño de bytes a escribir
86                                   &bytesEscritos, // tamaño de bytes escrit
87                                   NULL);
88
89         if(!escribir)
90         {
91             perror(destino);
92             exit(EXIT_FAILURE);
93         }
94         //memset(contenido, '\0', (int)bytesEscritos);
95         printf("%s ..... COPIADO", destino);
96         printf("\n\n");
97     }
98     free(contenido);
99     CloseHandle(o);
100    CloseHandle(d);
101 }
102
103 //FUNCION PRINCIPAL HILO
104 //Funciones que ejecutaran los hilos creados
105 DWORD WINAPI hilo(LPVOID lpParam)
106 {
107     char* buscar=(char*)calloc(4000,sizeof(char));
108     HANDLE h; //Handle para manejar los directorios leidos
109
110     //Nos sirve para leer todos los directorios contenidos en una ruta
111     WIN32_FIND_DATA wfd;
112     int i = 0, t = 0, num = 0;
113
114     //Utilizamos la estructura que creamos
115     RUTA *directorios, *raices = (RUTA*)lpParam;
116
117     //Creamos un hilo
118     DWORD *id_hilo = (DWORD*)calloc(1, sizeof(DWORD));
119     HANDLE *manHilo = (HANDLE*)calloc(1, sizeof(HANDLE));
120
121     //Concatenamos a una cadena limpia para evitar errores
122     strcpy (buscar, raices->Origen);
123     strcat (buscar, "\\*.");
124
125     //Leemos la ruta origen en busca de directorios
126     h = FindFirstFile (buscar, &wfd);
127
128     if(h == INVALID_HANDLE_VALUE)
129     {
130         perror(raices->Origen);
131         exit(-1);
132     }
133     else
134     {
135         do{ num++; } //Aqui primero contamos cuantos directorios hay

```

```

136     while(FindNextFile (h, &wfd));
137
138     //Reasignamos memoria para guardar los directorios encontrados
139     directorios = (RUTA*)calloc(num, sizeof(RUTA));
140     //Nuevamente leemos la ruta origen en busca de directorios
141     h = FindFirstFile (buscar, &wfd);
142
143     do
144     {
145         //Vamos construyendo las rutas de los directorios origen
146         concatenar(directorios[i].Origen, raices->Origen, wfd.cFileName);
147         //wfd es una especie de estructura que posee toda la info de los directorios
148         if(esDirectorio(wfd))
149         {
150             //wfd.cFileName es el nombre de cada directorio/archivo
151             if(ignorarPuntos(wfd.cFileName))
152             {
153                 printf("Soy el hilo %d. Encontre: %s", t, directorios[i].Origen);
154                 printf("\n");
155
156                 //Solo nos quedamos con directorios y construimos las rutas de destino
157                 concatenar(directorios[i].Destino, raices->Destino, wfd.cFileName);
158                 printf("Soy el hilo %d. Creare: %s", t, directorios[i].Destino);
159                 printf("\n\n");
160
161                 //Creamos los directorios destino
162                 if(!CreateDirectory(directorios[i].Destino, NULL))
163                 {
164                     perror(directorios[i].Destino);
165                     exit(-1);
166                 }
167                 //Vamos creando un nuevo hilo por cada directorio encontrado
168                 id_hilo = (DWORD*)realloc(id_hilo, sizeof(DWORD)*(t+1));
169                 manHilo = (HANDLE*)realloc(manHilo, sizeof(HANDLE)*(t+1));
170                 manHilo[t] = CreateThread(NULL, 0, hilo, &directorios[i], 0, &id_hilo[t]);
171                 t++;
172             }
173         }
174         else
175         {
176             //Ahora, creamos las rutas para los archivos dentro de cada directorio
177             concatenar(directorios[i].Destino, raices->Destino, wfd.cFileName);
178             printf("\n");
179             printf("Copiare el archivo: %s", directorios[i].Origen);
180             printf("\n");
181             //Leemos y escribimos los archivos
182             copiarArchivo(directorios[i].Origen, directorios[i].Destino);
183         }
184         i++;
185     }
186     while (FindNextFile (h, &wfd)); //Leemos hasta que no existan directorios
187     //Esperamos a que todos los hilos acaben
188     WaitForMultipleObjects(t, manHilo, TRUE, INFINITE);
189
190     FindClose (h);
191     free(directorios);
192     CloseHandle(manHilo);
193 }
194 }
195
196 int main(int argc, char *argv[])
197 {
198     //Creamos las rutas origen y destino
199     RUTA rutas;
200     DWORD id_hilo;
201     HANDLE manHilo;
202     WIN32_FIND_DATA wdata;
203
204     //Si no se ingresaron rutas por linea de comandos, tomamos las predeterminadas

```

```
205     if(argc != 3)
206     {
207         strcpy(rutas.Origen, "C:\\Users\\YaKerTaker\\Desktop\\origen");
208         strcpy(rutas.Destino, "C:\\Users\\YaKerTaker\\Desktop\\destino");
209     }
210     else
211     {
212         strcpy(rutas.Origen, argv[1]);
213         strcpy(rutas.Destino, argv[2]);
214     }
215
216     printf("Ruta Origen: %s", rutas.Origen);
217     printf("\n");
218     printf("Ruta Destino: %s", rutas.Destino);
219     printf("\n\n\n");
220
221     HANDLE handl = FindFirstFile (rutas.Origen, &wdata);
222     if(handl == INVALID_HANDLE_VALUE)
223     {
224         perror(rutas.Origen);
225         exit(-1);
226     }
227
228     /*Revisamos la ruta ingresada es un directorio, y que los directorios de origen
229     no existan en la ruta destino*/
230     if(esDirectorio(wdata))
231     {
232         if(!CreateDirectory(rutas.Destino, NULL))
233         {
234             perror(rutas.Destino);
235             exit(-1);
236         }
237         else
238         {
239             //Creamos un hilo y enviamos las rutas
240             manHilo = CreateThread(NULL, 0, hilo, &rutas, 0, &id_hilo);
241             WaitForSingleObject(manHilo, INFINITE);
242         }
243     }
244     return 0;
245 }
```



```
He creado 3 / 5 hilos id = 139900071474944
    Hijo Terminal Practica 5
He creado 4 / 5 hilos id = 139900071474944
    Hijo Terminal Practica 5
He creado 5 / 5 hilos id = 139900071474944
    Hijo Terminal Practica 5
He creado 3 / 10 hilos id = 139900079867648
    He creado 1 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 2 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 3 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 4 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 5 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
He creado 4 / 10 hilos id = 139900079867648
    He creado 1 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 2 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 3 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 4 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 5 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
He creado 5 / 10 hilos id = 139900079867648
    He creado 1 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 2 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 3 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 4 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
    He creado 5 / 5 hilos id = 139900071474944
        Hijo Terminal Practica 5
```





```

Hijo Terminal Practica 5
He creado 4 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 5 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 8 / 10 hilos id = 139900079867648
Hijo Terminal Practica 5
He creado 1 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 2 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 3 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 4 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 5 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 9 / 10 hilos id = 139900079867648
Hijo Terminal Practica 5
He creado 1 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 2 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 3 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 4 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 5 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 10 / 10 hilos id = 139900079867648
Hijo Terminal Practica 5
He creado 1 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 2 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 3 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 4 / 5 hilos id = 139900071474944
Hijo Terminal Practica 5
He creado 5 / 5 hilos id = 139900071474944
princess@princesa-pc:~/Escritorio$

```

## ✓ Programa 6.- Operaciones con matrices utilizando creación de hilos

```

enrike@enrike:~/Escritorio$ gcc 6.c tiempo.o -lpthread -o 6
enrike@enrike:~/Escritorio$ ./6
Ingrese el nuevo directorio: /home/enrike/Escritorio/resultados
MATRIZ 1
8.000 0.000 4.000 9.000 2.000 7.000 0.000 5.000 2.000 4.000
7.000 6.000 0.000 2.000 10.000 1.000 3.000 9.000 2.000 0.000
9.000 9.000 1.000 5.000 2.000 4.000 8.000 6.000 2.000 3.000
1.000 8.000 1.000 3.000 7.000 1.000 10.000 7.000 4.000 10.000
9.000 0.000 3.000 4.000 1.000 2.000 4.000 4.000 10.000 6.000
2.000 6.000 3.000 4.000 9.000 5.000 6.000 6.000 9.000 6.000
7.000 0.000 4.000 8.000 3.000 9.000 7.000 1.000 3.000 1.000
9.000 1.000 1.000 0.000 4.000 0.000 0.000 8.000 2.000 8.000
3.000 5.000 3.000 4.000 7.000 10.000 8.000 0.000 6.000 6.000
6.000 0.000 6.000 8.000 9.000 8.000 6.000 5.000 7.000 7.000

MATRIZ 2
6.000 3.000 9.000 6.000 3.000 2.000 6.000 2.000 10.000 9.000
10.000 0.000 1.000 1.000 3.000 6.000 0.000 0.000 6.000 4.000
6.000 1.000 5.000 2.000 8.000 1.000 8.000 3.000 4.000 2.000
9.000 9.000 5.000 5.000 4.000 7.000 7.000 8.000 9.000 4.000
4.000 8.000 2.000 3.000 9.000 5.000 9.000 8.000 5.000 2.000
10.000 1.000 4.000 2.000 1.000 10.000 3.000 7.000 2.000 8.000
9.000 9.000 6.000 3.000 3.000 8.000 10.000 10.000 3.000 8.000
1.000 8.000 4.000 2.000 9.000 0.000 5.000 8.000 8.000 0.000

1.000 8.000 4.000 2.000 9.000 0.000 5.000 8.000 8.000 0.000
10.000 6.000 10.000 1.000 8.000 9.000 0.000 1.000 5.000 1.000
7.000 3.000 10.000 0.000 4.000 3.000 8.000 4.000 0.000 0.000

----- Soy el hilo calculando la INVERSA
----- Soy el hilo calculando la TRANSPUESTA
----- Soy el hilo calculando la MULTIPLICACION
----- Soy el hilo calculando la RESTA
----- Soy el hilo calculando la SUMA
----- Soy el hilo leyendo RESULTADOS -----
SUMA
14.000 3.000 13.000 15.000 5.000 9.000 6.000 7.000 12.000 13.000
17.000 6.000 7.000 3.000 13.000 7.000 3.000 9.000 8.000 4.000
15.000 10.000 6.000 7.000 10.000 5.000 16.000 9.000 6.000 5.000
10.000 17.000 6.000 8.000 11.000 8.000 17.000 15.000 13.000 14.000
13.000 8.000 5.000 7.000 10.000 7.000 13.000 12.000 15.000 8.000
12.000 7.000 7.000 6.000 10.000 15.000 9.000 13.000 11.000 14.000
16.000 9.000 10.000 11.000 6.000 17.000 17.000 11.000 6.000 9.000
10.000 9.000 5.000 2.000 13.000 0.000 5.000 16.000 10.000 8.000
13.000 11.000 13.000 5.000 15.000 19.000 8.000 1.000 11.000 7.000
13.000 3.000 16.000 8.000 13.000 11.000 14.000 9.000 7.000 7.000

RESTA
2.000 -3.000 -5.000 3.000 -1.000 5.000 -6.000 3.000 -8.000 -5.000
-3.000 6.000 5.000 1.000 7.000 -5.000 3.000 9.000 -4.000 -4.000
3.000 8.000 -4.000 3.000 -6.000 3.000 0.000 3.000 -2.000 1.000
-8.000 -1.000 -4.000 -2.000 3.000 -6.000 3.000 -1.000 -5.000 6.000
5.000 -8.000 1.000 1.000 -8.000 -3.000 -5.000 -4.000 5.000 4.000
-8.000 5.000 -1.000 2.000 8.000 -5.000 3.000 -1.000 7.000 -2.000
-2.000 -9.000 -2.000 5.000 0.000 1.000 -3.000 -9.000 0.000 -7.000
8.000 -7.000 -3.000 -2.000 -5.000 0.000 -5.000 0.000 -6.000 8.000
-7.000 -1.000 -7.000 3.000 -1.000 1.000 8.000 -1.000 1.000 5.000
-1.000 -3.000 -4.000 8.000 5.000 5.000 -2.000 1.000 7.000 7.000

MULTIPLICAR
284.000 196.000 249.000 133.000 194.000 193.000 239.000 223.000 251.000 178.000
262.000 237.000 207.000 131.000 292.000 172.000 272.000 239.000 291.000 161.000
362.000 234.000 262.000 142.000 210.000 249.000 261.000 247.000 293.000 241.000
364.000 288.000 283.000 102.000 276.000 263.000 316.000 292.000 232.000 161.000
314.000 222.000 326.000 117.000 230.000 214.000 229.000 187.000 241.000 163.000
404.000 296.000 307.000 120.000 318.000 313.000 298.000 293.000 270.000 179.000
341.000 222.000 251.000 143.000 179.000 265.000 267.000 262.000 235.000 240.000
170.000 160.000 227.000 87.000 194.000 87.000 202.000 151.000 194.000 97.000
424.000 240.000 289.000 120.000 233.000 338.000 291.000 283.000 217.000 233.000
438.000 333.000 370.000 166.000 334.000 331.000 386.000 357.000 310.000 235.000

MULTIPLICAR
284.000 196.000 249.000 133.000 194.000 193.000 239.000 223.000 251.000 178.000
262.000 237.000 207.000 131.000 292.000 172.000 272.000 239.000 291.000 161.000
362.000 234.000 262.000 142.000 210.000 249.000 261.000 247.000 293.000 241.000
364.000 288.000 283.000 102.000 276.000 263.000 316.000 292.000 232.000 161.000
314.000 222.000 326.000 117.000 230.000 214.000 229.000 187.000 241.000 163.000
404.000 296.000 307.000 120.000 318.000 313.000 298.000 293.000 270.000 179.000
341.000 222.000 251.000 143.000 179.000 265.000 267.000 262.000 235.000 240.000
170.000 160.000 227.000 87.000 194.000 87.000 202.000 151.000 194.000 97.000
424.000 240.000 289.000 120.000 233.000 338.000 291.000 283.000 217.000 233.000
438.000 333.000 370.000 166.000 334.000 331.000 386.000 357.000 310.000 235.000

```

```

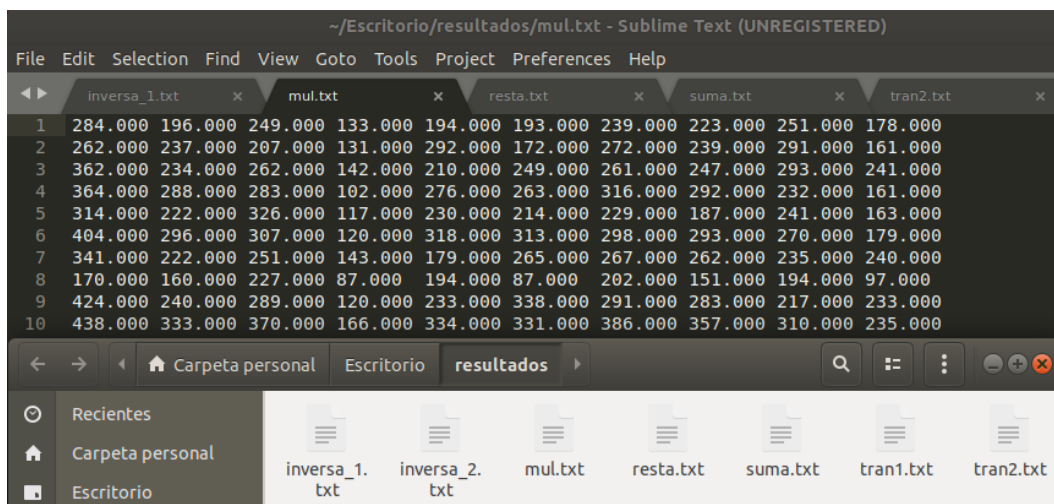
4.000 6.000 1.000 1.000 3.000 3.000 4.000 1.000 3.000 6.000 0.028 0.009 0.190 -0.065 0.008 -0.095 -0.293 -0.112 0.088 0.165
9.000 2.000 5.000 3.000 4.000 4.000 8.000 0.000 4.000 8.000 -0.423 0.382 -0.748 0.469 0.293 -0.101 0.487 -0.051 0.269 -0.997
2.000 10.000 2.000 7.000 1.000 9.000 3.000 4.000 7.000 9.000 -0.213 -0.228 0.670 -0.320 -0.144 -0.168 -0.667 -0.199 -0.179 1.035
7.000 1.000 4.000 1.000 2.000 5.000 9.000 0.000 10.000 8.000 -0.391 -0.215 0.698 -0.428 -0.222 -0.142 -0.675 -0.074 -0.130 1.140
0.000 3.000 8.000 10.000 4.000 6.000 7.000 0.000 8.000 6.000 0.271 0.132 -0.584 0.253 0.045 0.271 0.661 0.184 0.126 -0.950
5.000 9.000 6.000 7.000 4.000 6.000 1.000 8.000 0.000 5.000 -0.009 0.056 -0.214 0.174 0.046 0.042 0.349 0.051 -0.021 -0.318
2.000 2.000 2.000 4.000 10.000 9.000 3.000 2.000 6.000 7.000 0.327 0.194 -0.755 0.407 0.087 0.385 0.882 0.219 -0.031 -1.199
4.000 0.000 3.000 10.000 6.000 6.000 1.000 8.000 6.000 7.000 -0.028 -0.027 0.010 -0.056 0.059 0.115 -0.010 -0.019 -0.029 -0.010
0.059 -0.011 -0.013 0.065 0.033 -0.132 -0.131 -0.007 0.092 0.055

TRANSPUESTA MATRIZ 2
6.000 10.000 6.000 9.000 4.000 10.000 9.000 1.000 10.000 7.000 INVERSA MATRIZ 2
3.000 0.000 1.000 9.000 8.000 1.000 9.000 8.000 6.000 3.000 -0.159 -0.070 0.355 0.175 -0.338 -0.053 0.167 -0.004 0.070 -0.194
9.000 1.000 5.000 5.000 2.000 4.000 6.000 4.000 10.000 10.000 -0.050 -0.026 0.067 0.031 -0.112 -0.138 0.194 -0.003 0.080 -0.118
6.000 1.000 2.000 5.000 3.000 2.000 3.000 2.000 1.000 0.000 0.056 -0.041 -0.069 -0.031 -0.002 0.038 -0.050 0.027 0.021 0.087
3.000 3.000 8.000 4.000 9.000 1.000 3.000 9.000 8.000 4.000 -0.134 -0.468 0.586 0.401 -0.325 0.071 0.025 -0.166 0.186 -0.424
2.000 6.000 1.000 7.000 5.000 10.000 8.000 0.000 9.000 3.000 -0.046 -0.081 0.196 -0.035 -0.049 0.012 0.050 0.013 0.090 -0.151
6.000 0.000 8.000 7.000 9.000 3.000 10.000 5.000 0.000 8.000 0.129 0.130 -0.399 -0.124 0.410 0.079 -0.222 -0.074 -0.062 0.244
2.000 0.000 3.000 8.000 8.000 7.000 10.000 8.000 1.000 4.000 0.100 0.162 -0.246 -0.101 0.281 -0.048 -0.088 -0.062 -0.127 0.235
10.000 6.000 4.000 9.000 5.000 2.000 3.000 8.000 5.000 0.000 -0.075 -0.094 0.096 0.077 -0.140 0.126 -0.019 0.107 -0.023 -0.041
9.000 4.000 2.000 4.000 2.000 8.000 8.000 0.000 1.000 0.000 0.140 0.257 -0.387 -0.112 0.256 -0.002 -0.178 0.073 -0.144 0.268
0.048 0.034 0.015 -0.150 -0.034 -0.020 0.151 0.035 0.016 -0.085

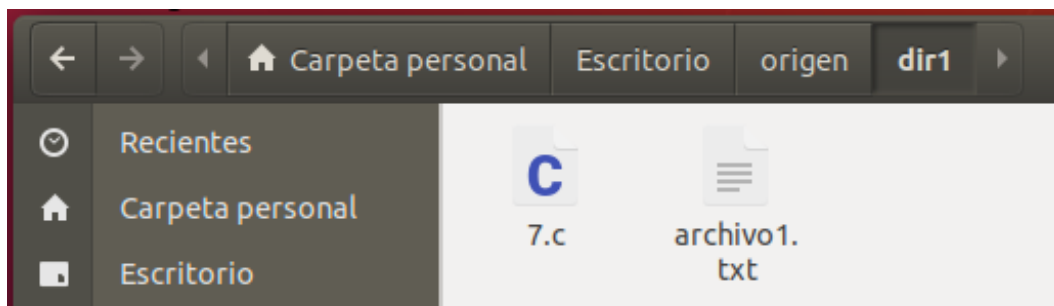
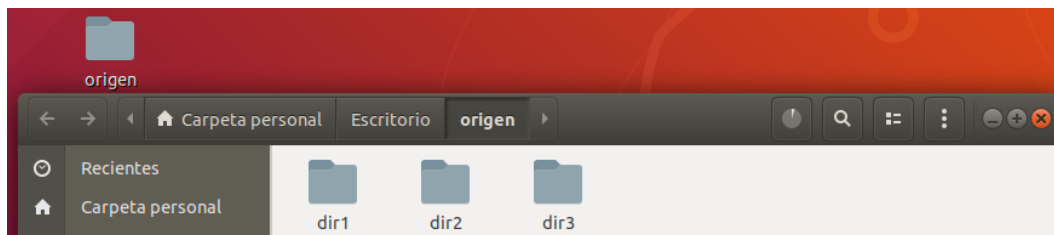
INVERSA MATRIZ 1
-0.213 -0.105 0.432 -0.262 -0.055 -0.165 -0.386 -0.022 -0.030 0.605
0.028 0.009 0.190 -0.065 0.008 -0.095 -0.293 -0.112 0.088 0.165

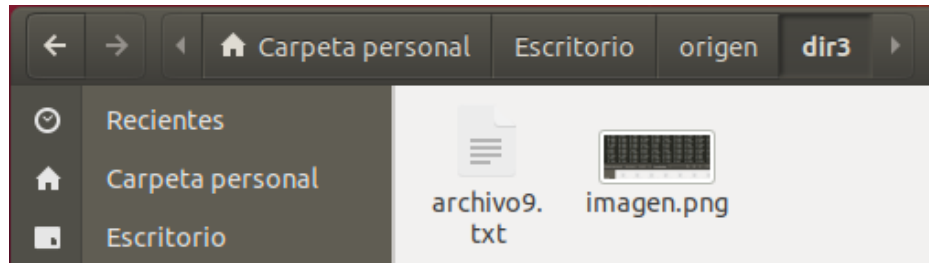
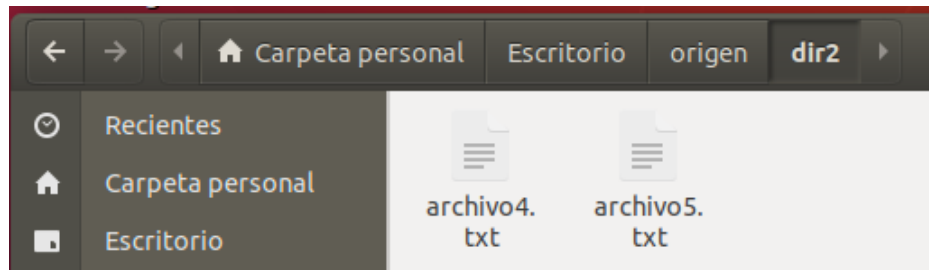
Tiempo ejecucion: 12.7274 s

```



- ✓ Programa 7.- Creación de directorios y copia de archivos concurrente utilizando creación de hilos





```
enrike@enrike:~/Escritorio$ gcc 7.c -lpthread -o 7
enrike@enrike:~/Escritorio$ ./7 /home/enrike/Escritorio/origen /home/enrike/Escritorio/destino
Ruta Origen: /home/enrike/Escritorio/origen
Ruta Destino: /home/enrike/Escritorio/destino

Soy el hilo 0. Encontre: /home/enrike/Escritorio/origen/dir3
Soy el hilo 0. Create: /home/enrike/Escritorio/destino/dir3

Soy el hilo 1. Encontre: /home/enrike/Escritorio/origen/dir2
Soy el hilo 1. Create: /home/enrike/Escritorio/destino/dir2

Soy el hilo 2. Encontre: /home/enrike/Escritorio/origen/dir1
Soy el hilo 2. Create: /home/enrike/Escritorio/destino/dir1

Copiare el archivo: /home/enrike/Escritorio/origen/dir1/archivo1.txt
/home/enrike/Escritorio/destino/dir1/archivo1.txt ..... COPIADO

Copiare el archivo: /home/enrike/Escritorio/origen/dir1/7.c
/home/enrike/Escritorio/destino/dir1/7.c ..... COPIADO

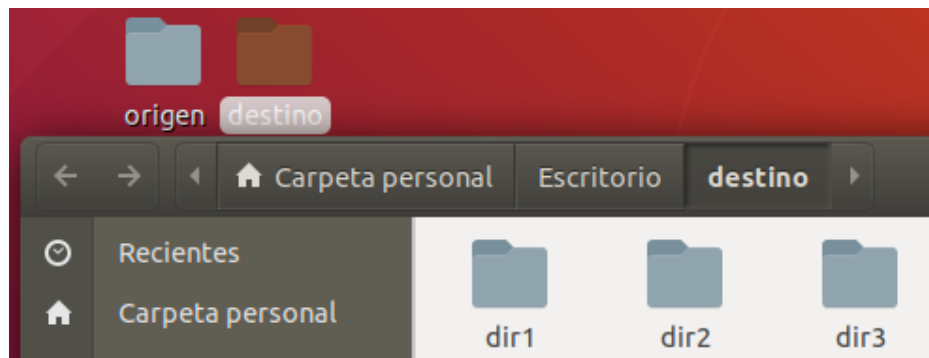
Copiare el archivo: /home/enrike/Escritorio/origen/dir2/archivo5.txt
/home/enrike/Escritorio/destino/dir2/archivo5.txt ..... COPIADO

Copiare el archivo: /home/enrike/Escritorio/origen/dir2/archivo4.txt
/home/enrike/Escritorio/destino/dir2/archivo4.txt ..... COPIADO

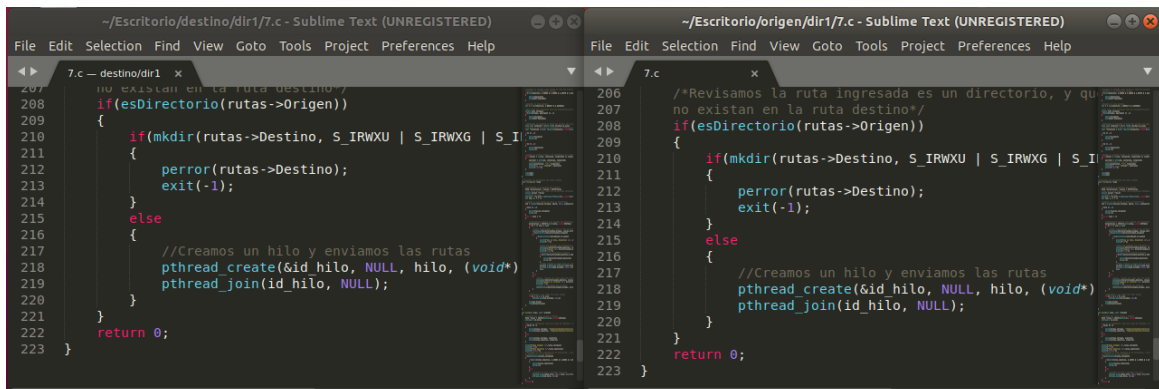
Copiare el archivo: /home/enrike/Escritorio/origen/dir3/imagen.png
/home/enrike/Escritorio/destino/dir3/imagen.png ..... COPIADO

Copiare el archivo: /home/enrike/Escritorio/origen/dir3/archivo9.txt
/home/enrike/Escritorio/destino/dir3/archivo9.txt ..... COPIADO

enrike@enrike:~/Escritorio$
```







### 2.3.2. Sección Windows:

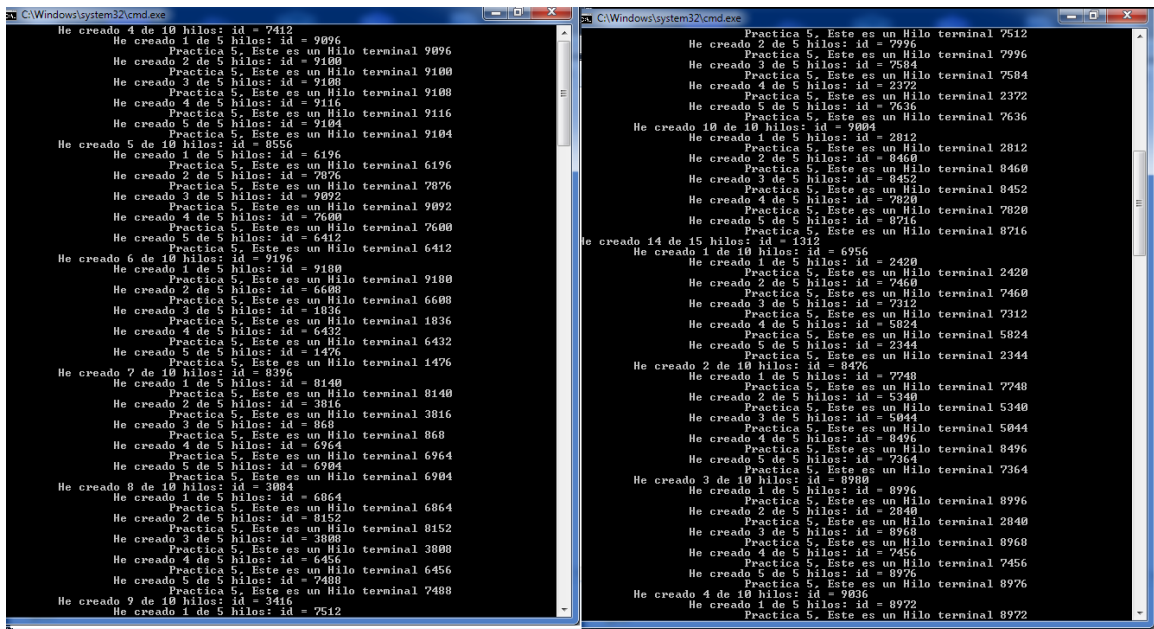
#### ✓ Programa 4: Creación de hilos en Windows

```
C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica5\Windows>gcc 4.c -o 4

C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica5\Windows>4
Valores al entrar al Hilo: 10 100
Valores a salir del Hilo: 20 200

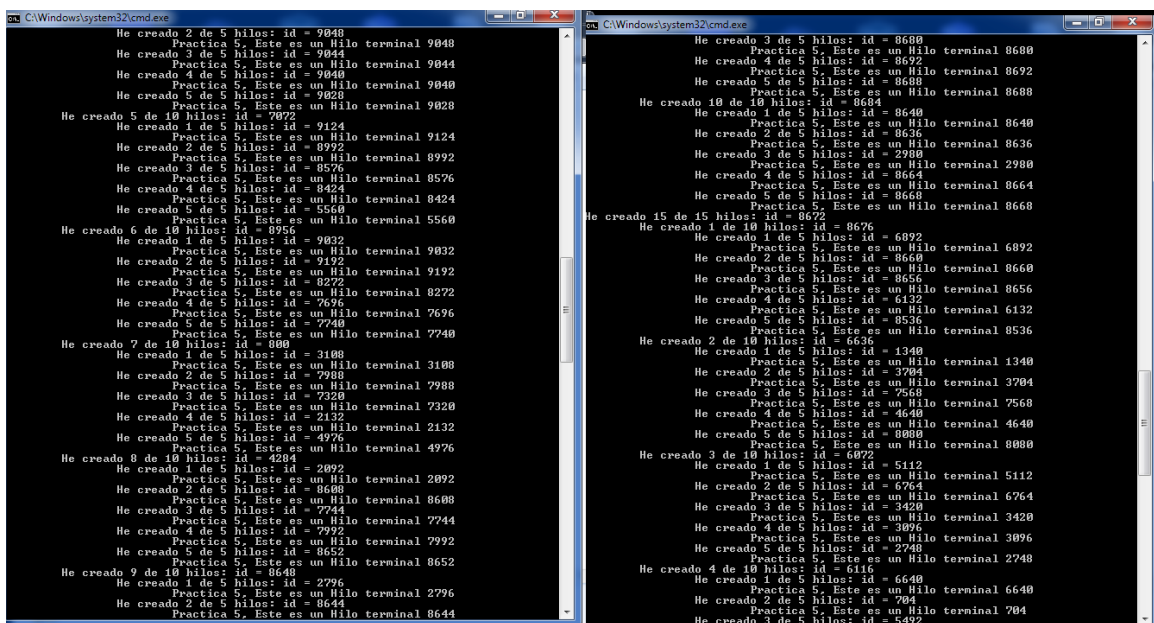
C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica5\Windows>
```

#### ✓ Programa 5.- Creación de hilos con procesos



```
C:\Windows\system32\cmd.exe
He creado 1 de 10 hilos: id = 7412
He creado 1 de 5 hilos: id = 9096
Practica 5, Este es un Hilo terminal 9096
He creado 2 de 5 hilos: id = 9100
Practica 5, Este es un Hilo terminal 9100
He creado 3 de 5 hilos: id = 9108
Practica 5, Este es un Hilo terminal 9108
He creado 4 de 5 hilos: id = 9116
Practica 5, Este es un Hilo terminal 9116
He creado 5 de 5 hilos: id = 9104
Practica 5, Este es un Hilo terminal 9104
He creado 6 de 10 hilos: id = 6556
He creado 1 de 5 hilos: id = 6196
Practica 5, Este es un Hilo terminal 6196
He creado 2 de 5 hilos: id = 7876
Practica 5, Este es un Hilo terminal 7876
He creado 3 de 5 hilos: id = 7092
Practica 5, Este es un Hilo terminal 7092
He creado 4 de 5 hilos: id = 7600
Practica 5, Este es un Hilo terminal 7600
He creado 5 de 5 hilos: id = 6412
Practica 5, Este es un Hilo terminal 6412
He creado 6 de 10 hilos: id = 9196
He creado 1 de 5 hilos: id = 9188
Practica 5, Este es un Hilo terminal 9188
He creado 2 de 5 hilos: id = 6608
Practica 5, Este es un Hilo terminal 6608
He creado 3 de 5 hilos: id = 1836
Practica 5, Este es un Hilo terminal 1836
He creado 4 de 5 hilos: id = 6432
Practica 5, Este es un Hilo terminal 6432
He creado 5 de 5 hilos: id = 1476
Practica 5, Este es un Hilo terminal 1476
He creado 7 de 10 hilos: id = 8396
He creado 1 de 5 hilos: id = 8140
Practica 5, Este es un Hilo terminal 8140
He creado 2 de 5 hilos: id = 3816
Practica 5, Este es un Hilo terminal 3816
He creado 3 de 5 hilos: id = 868
Practica 5, Este es un Hilo terminal 868
He creado 4 de 5 hilos: id = 6964
Practica 5, Este es un Hilo terminal 6964
He creado 5 de 5 hilos: id = 6904
Practica 5, Este es un Hilo terminal 6904
He creado 8 de 10 hilos: id = 3084
He creado 1 de 5 hilos: id = 6864
Practica 5, Este es un Hilo terminal 6864
He creado 2 de 5 hilos: id = 8152
Practica 5, Este es un Hilo terminal 8152
He creado 3 de 5 hilos: id = 3808
Practica 5, Este es un Hilo terminal 3808
He creado 4 de 5 hilos: id = 6456
Practica 5, Este es un Hilo terminal 6456
He creado 5 de 5 hilos: id = 7488
Practica 5, Este es un Hilo terminal 7488
He creado 9 de 10 hilos: id = 3416
He creado 1 de 5 hilos: id = 7512
Practica 5, Este es un Hilo terminal 7512

C:\Windows\system32\cmd.exe
Practica 5, Este es un Hilo terminal 7512
He creado 2 de 5 hilos: id = 7996
Practica 5, Este es un Hilo terminal 7996
He creado 3 de 5 hilos: id = 7584
Practica 5, Este es un Hilo terminal 7584
He creado 4 de 5 hilos: id = 2372
Practica 5, Este es un Hilo terminal 2372
He creado 5 de 5 hilos: id = 7636
Practica 5, Este es un Hilo terminal 7636
He creado 10 de 10 hilos: id = 9004
He creado 1 de 5 hilos: id = 2812
Practica 5, Este es un Hilo terminal 2812
He creado 2 de 5 hilos: id = 8460
Practica 5, Este es un Hilo terminal 8460
He creado 3 de 5 hilos: id = 8452
Practica 5, Este es un Hilo terminal 8452
He creado 4 de 5 hilos: id = 7820
Practica 5, Este es un Hilo terminal 7820
He creado 5 de 5 hilos: id = 8716
Practica 5, Este es un Hilo terminal 8716
He creado 14 de 15 hilos: id = 1312
He creado 1 de 5 hilos: id = 6956
Practica 5, Este es un Hilo terminal 6956
He creado 2 de 5 hilos: id = 2420
Practica 5, Este es un Hilo terminal 2420
He creado 3 de 5 hilos: id = 7460
Practica 5, Este es un Hilo terminal 7460
He creado 4 de 5 hilos: id = 7312
Practica 5, Este es un Hilo terminal 7312
He creado 5 de 5 hilos: id = 5824
Practica 5, Este es un Hilo terminal 5824
He creado 6 de 5 hilos: id = 2344
Practica 5, Este es un Hilo terminal 2344
He creado 2 de 10 hilos: id = 8476
He creado 1 de 5 hilos: id = 7740
Practica 5, Este es un Hilo terminal 7740
He creado 2 de 5 hilos: id = 5340
Practica 5, Este es un Hilo terminal 5340
He creado 3 de 5 hilos: id = 5044
Practica 5, Este es un Hilo terminal 5044
He creado 4 de 5 hilos: id = 8496
Practica 5, Este es un Hilo terminal 8496
He creado 5 de 5 hilos: id = 7364
Practica 5, Este es un Hilo terminal 7364
He creado 3 de 10 hilos: id = 8980
Practica 5, Este es un Hilo terminal 8996
He creado 4 de 5 hilos: id = 2840
Practica 5, Este es un Hilo terminal 2840
He creado 3 de 5 hilos: id = 8968
Practica 5, Este es un Hilo terminal 8968
He creado 4 de 5 hilos: id = 7456
Practica 5, Este es un Hilo terminal 7456
He creado 5 de 5 hilos: id = 8976
Practica 5, Este es un Hilo terminal 8976
He creado 4 de 10 hilos: id = 9036
He creado 1 de 5 hilos: id = 8972
Practica 5, Este es un Hilo terminal 8972
```



```
C:\Windows\system32\cmd.exe
He creado 2 de 5 hilos: id = 9840
Practica 5, Este es un Hilo terminal 9840
He creado 3 de 5 hilos: id = 9844
Practica 5, Este es un Hilo terminal 9844
He creado 4 de 5 hilos: id = 9840
Practica 5, Este es un Hilo terminal 9840
He creado 5 de 5 hilos: id = 9828
Practica 5, Este es un Hilo terminal 9828
He creado 5 de 10 hilos: id = 7072
He creado 1 de 5 hilos: id = 9124
Practica 5, Este es un Hilo terminal 9124
He creado 2 de 5 hilos: id = 8992
Practica 5, Este es un Hilo terminal 8992
He creado 3 de 5 hilos: id = 8576
Practica 5, Este es un Hilo terminal 8576
He creado 4 de 5 hilos: id = 8424
Practica 5, Este es un Hilo terminal 8424
He creado 5 de 5 hilos: id = 5560
Practica 5, Este es un Hilo terminal 5560
He creado 6 de 10 hilos: id = 8936
He creado 1 de 5 hilos: id = 9832
Practica 5, Este es un Hilo terminal 9832
He creado 2 de 5 hilos: id = 9192
Practica 5, Este es un Hilo terminal 9192
He creado 3 de 5 hilos: id = 8272
Practica 5, Este es un Hilo terminal 8272
He creado 4 de 5 hilos: id = 7696
Practica 5, Este es un Hilo terminal 7696
He creado 5 de 5 hilos: id = 7740
Practica 5, Este es un Hilo terminal 7740
He creado 7 de 10 hilos: id = 8080
He creado 1 de 5 hilos: id = 3108
Practica 5, Este es un Hilo terminal 3108
He creado 2 de 5 hilos: id = 7988
Practica 5, Este es un Hilo terminal 7988
He creado 3 de 5 hilos: id = 7320
Practica 5, Este es un Hilo terminal 7320
He creado 4 de 5 hilos: id = 2132
Practica 5, Este es un Hilo terminal 2132
He creado 5 de 5 hilos: id = 4976
Practica 5, Este es un Hilo terminal 4976
He creado 8 de 10 hilos: id = 4284
He creado 1 de 5 hilos: id = 2092
Practica 5, Este es un Hilo terminal 2092
He creado 2 de 5 hilos: id = 8608
Practica 5, Este es un Hilo terminal 8608
He creado 3 de 5 hilos: id = 7744
Practica 5, Este es un Hilo terminal 7744
He creado 4 de 5 hilos: id = 7992
Practica 5, Este es un Hilo terminal 7992
He creado 5 de 5 hilos: id = 8652
Practica 5, Este es un Hilo terminal 8652
He creado 9 de 10 hilos: id = 8648
He creado 1 de 5 hilos: id = 2796
Practica 5, Este es un Hilo terminal 2796
He creado 2 de 5 hilos: id = 8644
Practica 5, Este es un Hilo terminal 8644

C:\Windows\system32\cmd.exe
He creado 3 de 5 hilos: id = 8680
Practica 5, Este es un Hilo terminal 8680
He creado 4 de 5 hilos: id = 8692
Practica 5, Este es un Hilo terminal 8692
He creado 5 de 5 hilos: id = 8688
Practica 5, Este es un Hilo terminal 8688
He creado 10 de 10 hilos: id = 8684
He creado 1 de 5 hilos: id = 8640
Practica 5, Este es un Hilo terminal 8640
He creado 2 de 5 hilos: id = 8636
Practica 5, Este es un Hilo terminal 8636
He creado 3 de 5 hilos: id = 2980
Practica 5, Este es un Hilo terminal 2980
He creado 4 de 5 hilos: id = 8664
Practica 5, Este es un Hilo terminal 8664
He creado 5 de 5 hilos: id = 8660
Practica 5, Este es un Hilo terminal 8660
He creado 15 de 15 hilos: id = 8672
He creado 1 de 10 hilos: id = 8676
Practica 5, Este es un Hilo terminal 6892
He creado 2 de 5 hilos: id = 8660
Practica 5, Este es un Hilo terminal 8660
He creado 3 de 5 hilos: id = 8656
Practica 5, Este es un Hilo terminal 8656
He creado 4 de 5 hilos: id = 6132
Practica 5, Este es un Hilo terminal 6132
He creado 5 de 5 hilos: id = 8536
Practica 5, Este es un Hilo terminal 8536
He creado 2 de 10 hilos: id = 6636
He creado 1 de 5 hilos: id = 1340
Practica 5, Este es un Hilo terminal 1340
He creado 2 de 5 hilos: id = 3704
Practica 5, Este es un Hilo terminal 3704
He creado 3 de 5 hilos: id = 7568
Practica 5, Este es un Hilo terminal 7568
He creado 4 de 5 hilos: id = 4640
Practica 5, Este es un Hilo terminal 4640
He creado 5 de 5 hilos: id = 8800
Practica 5, Este es un Hilo terminal 8800
He creado 3 de 10 hilos: id = 6072
He creado 1 de 5 hilos: id = 5112
Practica 5, Este es un Hilo terminal 5112
He creado 2 de 5 hilos: id = 6764
Practica 5, Este es un Hilo terminal 6764
He creado 3 de 5 hilos: id = 3420
Practica 5, Este es un Hilo terminal 3420
He creado 4 de 5 hilos: id = 3096
Practica 5, Este es un Hilo terminal 3096
He creado 5 de 5 hilos: id = 2740
Practica 5, Este es un Hilo terminal 2740
He creado 4 de 10 hilos: id = 6116
He creado 1 de 5 hilos: id = 6640
Practica 5, Este es un Hilo terminal 6640
He creado 2 de 5 hilos: id = 704
Practica 5, Este es un Hilo terminal 704
He creado 3 de 5 hilos: id = 5492
Practica 5, Este es un Hilo terminal 5492
```



```

-1.000 -6.000 7.000 -4.000 -7.000 7.000 0.000 6.000 -1.000 -5.000
-1.000 -1.000 4.000 5.000 -2.000 -2.000 -5.000 -1.000 3.000 2.000
0.000 7.000 -8.000 3.000 -1.000 -2.000 -1.000 -6.000 -2.000
-3.000 -8.000 0.000 4.000 -3.000 -7.000 -4.000 -2.000 -4.000 0.000
2.000 -4.000 -7.000 -9.000 -1.000 0.000 -1.000 -5.000 9.000
-2.000 3.000 -6.000 2.000 3.000 2.000 -2.000 2.000 0.000 -1.000
0.000 8.000 5.000 1.000 5.000 8.000 4.000 -4.000 -5.000 -3.000

MULTIPLICAR
400.000 338.000 334.000 451.000 373.000 330.000 468.000 437.000 487.000 336.000
354.000 277.000 250.000 333.000 332.000 239.000 305.000 300.000 354.000 287.000
341.000 325.000 305.000 314.000 286.000 328.000 426.000 322.000 400.000 333.000
304.000 269.000 311.000 341.000 226.000 269.000 289.000 338.000 401.000 239.000
332.000 265.000 223.000 294.000 289.000 212.000 369.000 288.000 362.000 350.000
329.000 348.000 322.000 306.000 287.000 281.000 373.000 343.000 398.000 289.000
297.000 238.000 206.000 295.000 257.000 166.000 271.000 270.000 328.000 272.000
268.000 255.000 274.000 299.000 150.000 219.000 300.000 353.000 392.000 216.000
299.000 283.000 293.000 244.000 202.000 189.000 300.000 285.000 384.000 205.000
467.000 433.000 398.000 453.000 401.000 380.000 521.000 470.000 546.000 403.000

TRANSPUESTA MATRIZ 1
8.000 1.000 8.000 5.000 6.000 1.000 5.000 8.000 5.000 9.000
9.000 5.000 2.000 1.000 2.000 10.000 2.000 5.000 10.000 9.000
7.000 8.000 5.000 9.000 4.000 1.000 6.000 2.000 3.000 7.000
7.000 6.000 4.000 1.000 10.000 8.000 9.000 1.000 2.000 7.000
10.000 8.000 10.000 3.000 7.000 4.000 3.000 0.000 3.000 8.000
9.000 3.000 9.000 8.000 5.000 8.000 2.000 6.000 3.000 8.000
1.000 8.000 9.000 7.000 4.000 8.000 3.000 2.000 10.000
8.000 6.000 1.000 8.000 1.000 7.000 5.000 8.000 4.000 5.000
0.000 0.000 8.000 3.000 5.000 3.000 2.000 4.000 10.000 5.000

TRANSPUESTA MATRIZ 2
2.000 8.000 9.000 8.000 5.000 8.000 2.000 6.000 7.000 1.000
7.000 8.000 2.000 7.000 3.000 3.000 10.000 9.000 7.000 1.000
3.000 8.000 7.000 2.000 0.000 9.000 6.000 9.000 9.000 2.000
8.000 6.000 10.000 5.000 5.000 5.000 10.000 0.000 6.000
2.000 8.000 10.000 10.000 9.000 1.000 6.000 2.000 0.000 3.000
8.000 4.000 1.000 1.000 7.000 9.000 9.000 7.000 1.000 0.000
10.000 9.000 1.000 7.000 9.000 10.000 7.000 2.000 4.000 6.000
8.000 9.000 5.000 2.000 2.000 8.000 7.000 9.000 2.000 9.000
5.000 10.000 10.000 4.000 2.000 9.000 6.000 9.000 10.000 10.000
2.000 2.000 3.000 10.000 7.000 5.000 8.000 1.000 5.000 10.000

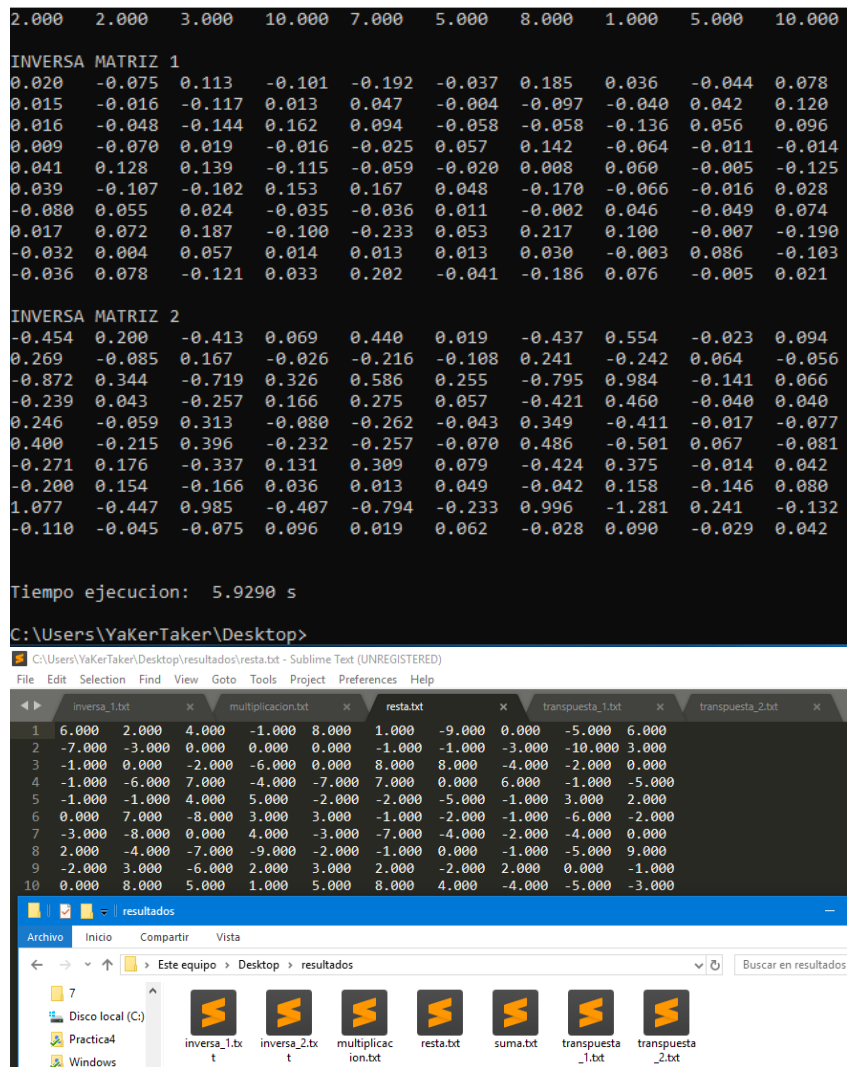
INVERSA MATRIZ 1
0.020 -0.075 0.113 -0.101 -0.192 -0.037 0.185 0.036 -0.044 0.078
0.015 -0.016 -0.117 0.013 0.047 -0.004 -0.097 -0.040 0.042 0.120
0.016 -0.048 -0.144 0.162 0.094 -0.058 -0.058 -0.136 0.056 0.096
0.009 -0.070 0.019 -0.016 -0.025 0.057 0.142 -0.064 -0.011 -0.014
0.041 0.128 0.139 -0.115 -0.059 -0.020 0.008 0.060 -0.005 -0.125
0.039 -0.107 -0.102 0.153 0.167 0.048 -0.170 -0.066 -0.016 0.028
-0.080 0.055 0.024 -0.035 -0.036 0.011 -0.002 0.046 -0.049 0.074
0.017 0.072 0.187 -0.100 -0.233 0.053 0.217 0.100 -0.007 -0.190
-0.032 0.004 0.057 0.014 0.013 0.013 0.030 -0.003 0.086 -0.103
-0.036 0.078 -0.121 0.033 0.202 -0.041 -0.186 0.076 -0.005 0.021

INVERSA MATRIZ 2
-0.454 0.200 -0.413 0.069 0.440 0.019 -0.437 0.554 -0.023 0.094
0.269 -0.085 0.167 -0.026 -0.216 -0.108 0.241 -0.242 0.064 -0.056
-0.872 0.344 -0.719 0.326 0.586 0.255 -0.795 0.984 -0.141 0.066
-0.239 0.043 -0.257 0.166 0.275 0.057 -0.421 0.460 -0.040 0.040
0.246 -0.059 0.313 -0.080 -0.262 -0.043 0.349 -0.411 -0.017 -0.077
0.400 -0.215 0.396 -0.232 -0.257 -0.070 0.486 -0.501 0.067 -0.081
-0.271 0.176 -0.337 0.131 0.309 0.079 -0.424 0.375 -0.014 0.042
-0.200 0.154 -0.166 0.036 0.013 0.049 -0.042 0.158 -0.146 0.080
1.077 -0.447 0.985 -0.407 -0.794 -0.233 0.996 -1.281 0.241 -0.132
-0.110 -0.045 -0.075 0.096 0.019 0.062 -0.028 0.090 -0.029 0.042

Tiempo ejecucion: 5.9290 s

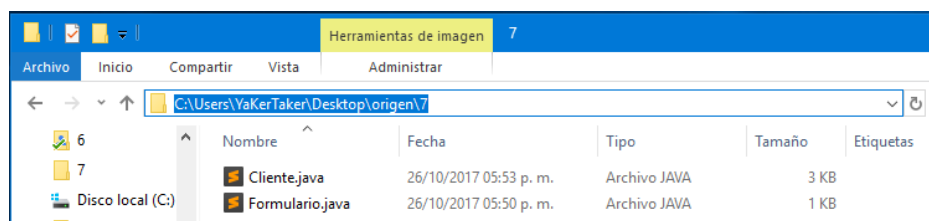
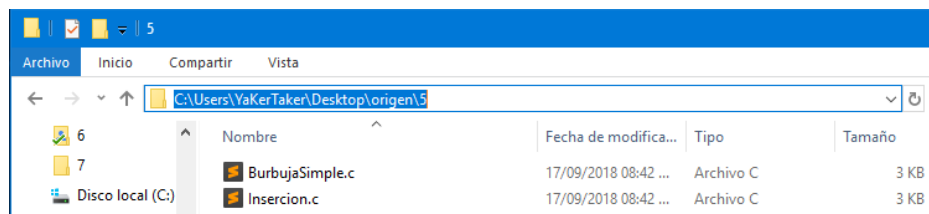
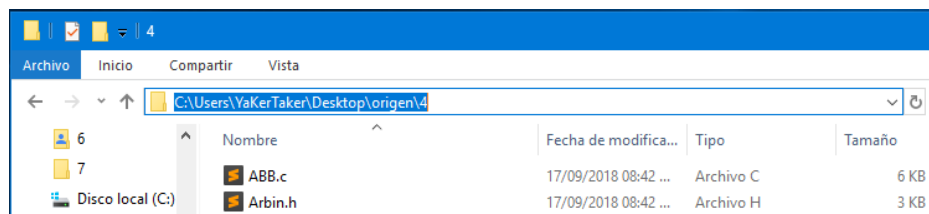
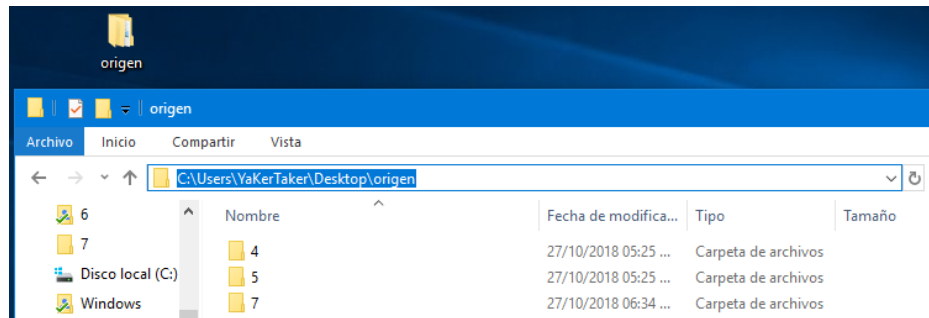
C:\Users\YaKerTaker\Desktop>

```





✓ **Programa 7.- Creación de directorios y copia de archivos concurrente utilizando creación de hilos**



```

C:\Users\YaKerTaker\Desktop>gcc 7.c -o 7

C:\Users\YaKerTaker\Desktop>7 C:\Users\YaKerTaker\Desktop\origen C:\Users\YaKerTaker\Desktop\destino
Ruta Origen: C:\Users\YaKerTaker\Desktop\origen
Ruta Destino: C:\Users\YaKerTaker\Desktop\destino

Soy el hilo 0. Encontre: C:\Users\YaKerTaker\Desktop\origen\4
Soy el hilo 0. Create: C:\Users\YaKerTaker\Desktop\destino\4

Soy el hilo 1. Encontre: C:\Users\YaKerTaker\Desktop\origen\5
Soy el hilo 1. Create: C:\Users\YaKerTaker\Desktop\destino\5

Copiare el archivo: C:\Users\YaKerTaker\Desktop\origen\4\ABB.c
Soy el hilo 2. Encontre: C:\Users\YaKerTaker\Desktop\origen\7
Soy el hilo 2. Create: C:\Users\YaKerTaker\Desktop\destino\7

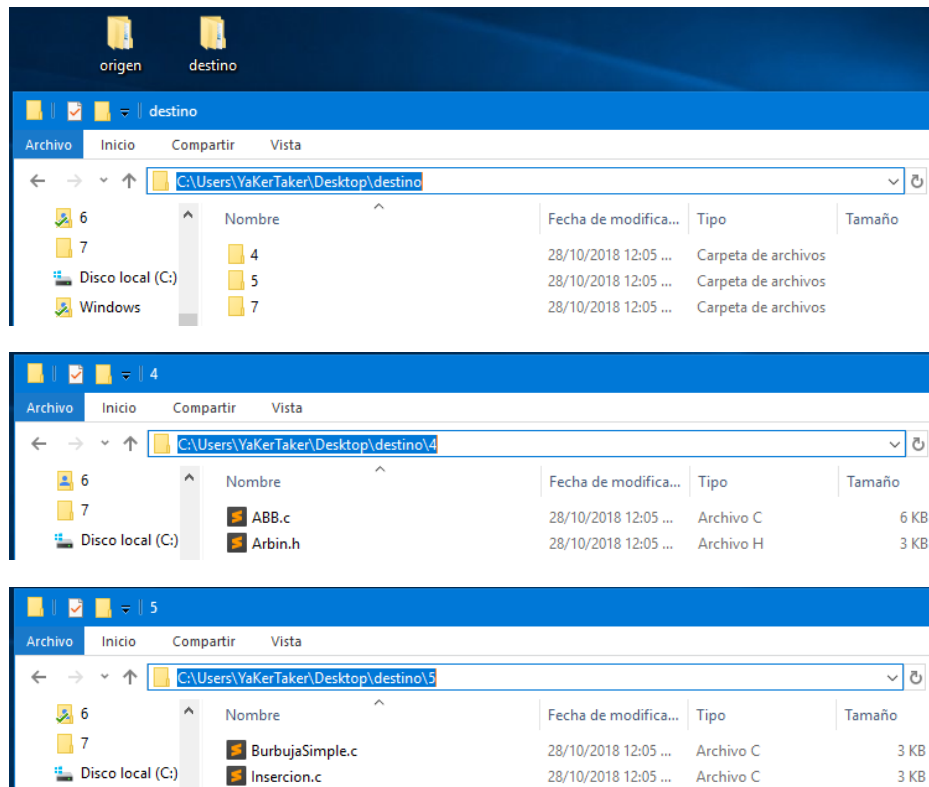
Copiare el archivo: C:\Users\YaKerTaker\Desktop\origen\5\BurbujaSimple.c
C:\Users\YaKerTaker\Desktop\destino\4\ABB.c ..... COPIADO

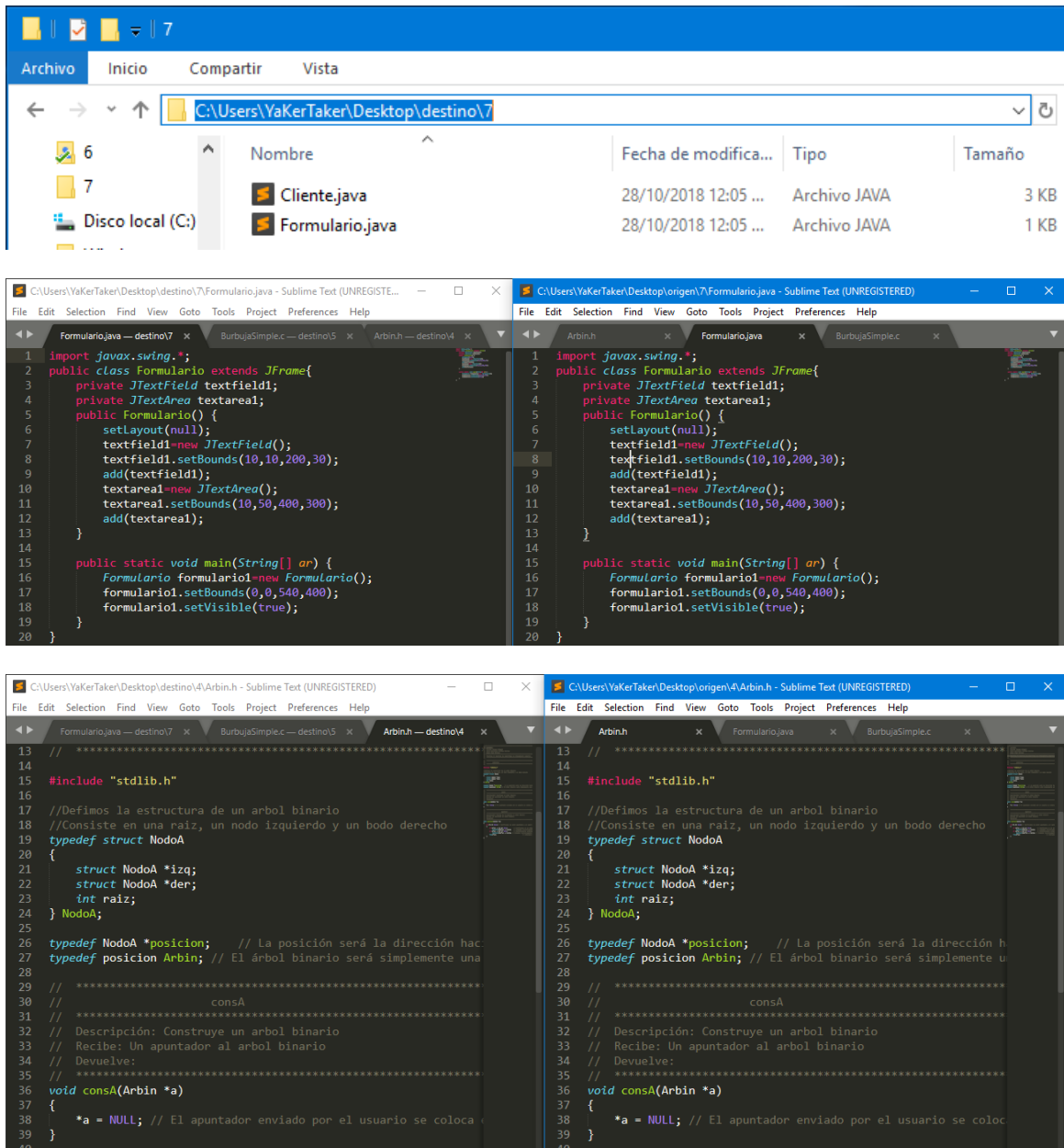
Copiare el archivo: C:\Users\YaKerTaker\Desktop\origen\7\Cliente.java
C:\Users\YaKerTaker\Desktop\destino\5\BurbujaSimple.c ..... COPIADO
C:\Users\YaKerTaker\Desktop\destino\7\Cliente.java ..... COPIADO

Copiare el archivo: C:\Users\YaKerTaker\Desktop\origen\4\Arbin.h
Copiare el archivo: C:\Users\YaKerTaker\Desktop\origen\5\Insercion.c
Copiare el archivo: C:\Users\YaKerTaker\Desktop\origen\7\Formulario.java
C:\Users\YaKerTaker\Desktop\destino\4\Arbin.h ..... COPIADO
C:\Users\YaKerTaker\Desktop\destino\5\Insercion.c ..... COPIADO
C:\Users\YaKerTaker\Desktop\destino\7\Formulario.java ..... COPIADO

C:\Users\YaKerTaker\Desktop>

```





### 3. Observaciones

- ✓ Para los programas desarrollados en Linux, se utiliza el siguiente comando para compilar: `gcc nombre_programa.c -lpthread -o nombre_programa`
- ✓ Algunas aplicaciones deben compilarse con atributos extras en el comando GCC y `lpthread`. La instrucción para compilar éstas aplicaciones se encuentra al inicio de su respectivo código.
- ✓ En el punto 3 de Linux, el segundo código de ejemplo para la creación de hilos en este sistema operativo, al momento de compilar con el comando de arriba, aparece un warning respecto al cast a tipo de variable `void` que se le hace al resultado del hilo cuando se regresa al `main`. A pesar de esto, la ejecución del programa es la esperada, por lo que podemos ignorarlo.

- ✓ En el punto 6 de Linux, el programa nos pide que ingresemos una ruta en donde se escribirán los archivos con los resultados de las operaciones con matrices. Ésta ruta debe ser ingresada de forma completa, por ejemplo:  
`\home\usuario\Escritorio\resultados`  
Esto aplica de igual manera para la entrada de las rutas en el punto 7.
- ✓ En las versiones de Windows de estos mismos programas, no es necesario ingresar la ruta completa.
- ✓ En este mismo punto, para la medición del tiempo de ejecución de la aplicación, se utilizaron los códigos de *tiempo.h* y *tiempo.c* de la práctica 4.
- ✓ Es importante que tanto el directorio en donde se escribirán los resultados de las operaciones con matrices en el punto 6 como el directorio de destino en el punto 7 no existan antes de la ejecución de ambos programas, ya que de existir, ambos programas retornarán un error y acabarán su ejecución.
- ✓ Análogo al punto anterior, es necesario que la ruta de origen en el punto 7 exista antes de ejecutar el código.
- ✓ Las rutas que se utilizan en los puntos 6 y 7 se aceptan como parámetros de entrada por medio de la línea de comandos
- ✓ En el punto 7, la versión de Linux acepta todo tipo de archivos para copiarlos, sin embargo, la versión de Linux tiene problemas con las imágenes, por lo que únicamente funciona con archivos que contienen texto.
- ✓ Mientras que en Linux se usaron las llamadas al sistema `scandir()`, `stat()` y la estructura `dirp()`, en Linux se utilizó las llamadas al sistema `FindFirstFile()`, `FindNextFile()`, y variables de tipo `WIN32_FIND_DATA`.

## 4. Análisis Crítico

La diferencia de la creación de múltiples hilos en los sistemas operativos Windows y Linux, es notable por la sintaxis que existen en ambos:

- Linux : Utilizamos las funciones de la biblioteca **pthread**, y el tipo de variable `pthread_t`.
- Windows: Utilizamos las funciones para manejo de hilos contenidas en la biblioteca **windows.h**, y los tipos de variable `DWORD` y `HANDLE`. Además, la función para un hilo es de tipo `DWORD WINAPI`.

La manera de manejar hilos en cada sistema operativo cambia en la sintaxis que se usa. En Linux se usa `pthread_create()` y `pthread_join()`, mientras que en Windows se usa `CreateThread()` por medio de un manejador del id de cada hilo utilizando una variable de tipo `HANDLE`; para esperar a

que todos los hilos terminen, utilizamos el mismo comando para procesos: `WaitForSingleObject()` ó `WaitForMultipleObjects()` para multihilos.

Al ejecutar el programa de las matrices, hicimos una comparativa de los tiempos en los que se realiza teniendo los siguientes resultados:

Sistemas Operativos	Forma	Tiempo
Linux	Secuencial	0.8645 seg
	<b>Con procesos</b>	<b>0.7959 seg</b>
	Hilos	12.7274 seg
Windows	Secuencial	8.2770 seg
	Con procesos	8.4970 seg
	<b>Hilos</b>	<b>5.9290 seg</b>

En el sistema operativo Windows, el usar hilos disminuye el tiempo respecto a la implementación secuencial y con creación de procesos, sin embargo en Linux no pasa eso, al contrario, el tiempo es exageradamente mayor con hilos que con procesos y en forma secuencial.

## 5. Conclusiones

La programación multihilos sin duda tiene mucho campo de aplicación, desde los sistemas operativos hasta en la tecnologías que usamos cotidianamente.

En la elaboración de este trabajo, después de discutir varios aspectos de la práctica nos dimos cuenta que obtuvimos conceptos que nos darán las bases para desarrollar aplicaciones en c en el sistema operativo tanto UNIX como Windows utilizando hilos, con la finalidad de poder realizar más de una tarea a la vez y administrando correctamente los recursos del equipo de cómputo.

La utilidad de la programación con hilos resulta evidente. Por ejemplo, cualquier navegador web (FireFox, Chrome, Opera) puede descargar un archivo de un sitio o página web, y acceder a otro sitio al mismo tiempo. Si el navegador puede realizar simultáneamente dos tareas, no tendrá que esperar hasta que el archivo haya terminado de descargarse para poder navegar a otro sitio.

En conclusión la programación multihilo esta presente en la mayor parte de las aplicaciones informáticas que usamos cotidianamente sin importar el lenguaje en que se desenvuelvan, el concepto de Thread o Hilo seguirá siendo el mismo