



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

PRACTICA 5

Comunicación interprocesos (IPC) en Linux y Windows

SISTEMAS OPERATIVOS

Profesor: Cortes Galicia Jorge

Grupo: 2CM9

Integrantes:

Beltran García Juan

Hernández Méndez Oliver Manuel

López López Rodrigo

Rangel Lozada Kevin Sebastián

INSTITUTO POLITÉCNICO NACIONAL



ESCOM



Índice

1	Introducción Teórica.....	1
2	Desarrollo Experimental	2
2.1	Ejercicio 1.....	2
2.2	Ejercicio 2.....	5
2.3	Ejercicio 3.....	6
2.4	Ejercicio 4.....	8
2.4.1	Codificación en Linux.....	8
2.4.2	Codificación en Windows.....	18
2.5	Ejercicio 5.....	24
2.6	Ejercicio 6.....	26
2.7	Ejercicio 7.....	29
2.7.1	Codificación en Linux.....	29
2.7.2	Codificación en Windows.....	35
3	Conclusiones.....	42

1 Introducción Teórica

Como hemos visto teóricamente y en otras prácticas, el manejo de la información entre procesos es algo fundamental e indispensable, hablando principalmente de la funcionalidad de los sistemas operativos y las aplicaciones que lo emplean.

Un proceso puede ser de dos tipos:

- Proceso independiente.
- Proceso de cooperación.

Un proceso independiente no se ve afectado por la ejecución de otros procesos, mientras que un proceso cooperativo puede verse afectado por otros procesos en ejecución. Aunque uno puede pensar que esos procesos, que se ejecutan de forma independiente, se ejecutarán de manera muy eficiente, en realidad, hay muchas situaciones en las que la naturaleza cooperativa se puede utilizar para aumentar la velocidad, la conveniencia y el modularidad computacional. La comunicación entre procesos (IPC) es un mecanismo que permite a los procesos comunicarse entre sí y sincronizar sus acciones. La comunicación entre estos procesos puede verse como un método de cooperación entre ellos. Los procesos pueden comunicarse entre sí a través de ambos:

1. Memoria compartida
2. Paso de mensajes

La IPC provee un mecanismo que permite a los procesos comunicarse y sincronizarse entre sí, normalmente a través de un sistema de bajo nivel de paso de mensajes que ofrece la red subyacente.

Un sistema operativo puede implementar ambos métodos de comunicación. Primero, discutiremos los métodos de comunicación de memoria compartida y luego el paso de mensajes. La comunicación entre procesos que utilizan memoria compartida requiere que los procesos compartan alguna variable y depende completamente de cómo la implementará el programador. Una forma de comunicación que utiliza la memoria compartida se puede imaginar así: Supongamos que proceso1 y proceso2 se están ejecutando simultáneamente y comparten algunos recursos o usan información de otro proceso. Process1 genera información sobre ciertos cálculos o recursos que se utilizan y la mantiene como un registro en la memoria compartida. Cuando process2 necesita usar la información compartida, verificará el registro almacenado en la memoria compartida y tomará nota de la información generada por process1 y actuará en consecuencia.

Ahora, comenzaremos nuestra discusión sobre la comunicación entre procesos a través del paso de mensajes. En este método, los procesos se comunican entre sí sin utilizar ningún tipo de memoria compartida. Si dos procesos p1 y p2 quieren comunicarse entre sí, proceden de la siguiente manera:

- Establezca un enlace de comunicación (si ya existe un enlace, no es necesario volver a establecerlo).
- Empiece a intercambiar mensajes utilizando primitivas básicas.

Necesitamos al menos dos primitivas:

- enviar (mensaje, destino) o enviar (mensaje)
- recibir (mensaje, host) o recibir (mensaje)

2 Desarrollo Experimental

2.1 Ejercicio 1

A través de la ayuda en línea que proporciona Linux, investigue el funcionamiento de la función: `pipe()`, `shmget()`, `shmat()`. Explique los argumentos y retorno de la función.

pipe()

Esta función se emplea para crear una tubería o pipe

Conceptualmente, un tubo o pipe es una conexión entre dos procesos, de manera que la salida estándar de un proceso se convierte en la entrada estándar del otro proceso. En el sistema operativo UNIX, las tuberías son útiles para la comunicación entre procesos relacionados,

Una tubería tiene en realidad dos descriptores de fichero: uno para el extremo de escritura y otro para el extremo de lectura. Como los descriptores de fichero de UNIX son simplemente enteros, un pipe o tubería no es más que un array de dos enteros:

Para crear la tubería se emplea la función `pipe()`, que abre dos descriptores de fichero y almacena su valor en los dos enteros que contiene el array de descriptores de fichero. El primer descriptor de fichero es abierto como `O_RDONLY`, es decir, sólo puede ser empleado para lecturas. El segundo se abre como `O_WRONLY`, limitando su uso a la escritura. De esta manera se asegura que el pipe sea de un solo sentido: por un extremo se escribe y por el otro se lee, pero nunca al revés. Ya hemos dicho que si se precisa una comunicación “full-duplex”, será necesario crear dos tuberías para ello.

Sintaxis:

```
int pipe(int fds[2]);
```

Parámetros:

`fds[0]` : Sera el descriptor de fichero para el extremo de lectura

`fds[1]` : Sera el descriptor de fichero para el extremo de escritura

Retorno:

Se retorna el entero 0 en caso de éxito y se retorna -1 en caso de algún error

shmget()

Esta función se emplea para la creación o acceso a una zona de memoria compartida.

La memoria compartida se crea por un proceso mediante una llamada al sistema, la zona que se reserva en memoria no está en el espacio de direcciones del proceso, es una zona de memoria gestionada por el sistema operativo. Después, otros procesos a los que se les dé permiso para acceder a esa zona de memoria podrán también leer o escribir de ella.

Sintaxis:

```
int shmget(key_t key, int size, int shmflg);
```

Parámetros:

key: Es una clave que genera el sistema y que será un elemento esencial para poder acceder a la zona de memoria que crearemos. La clave se obtiene previamente utilizando la función `ftok()`; que produce siempre una clave fija con los mismos argumentos: Para usarla:

```
key = ftok(".", 'S');
```

La clave debe ser la misma para todos los procesos que quieran usar esta zona de memoria

size: Indicará el tamaño de la memoria compartida. Se suele utilizar la opción de `sizeof (tipo_variable)` para que se tome el tamaño del tipo de dato que habrá en la memoria compartida.

shmflg: Indicará los derechos para acceder a la memoria, si se crea si no existe y caso de que exista, se da o no un error.

Retorno:

Se retorna el identificador de la memoria compartida si no ha habido error

Se retorna -1 en caso de un error

shmat()

Una vez ya creada la memoria compartida, pero para utilizarla necesitamos saber su dirección (observe que shmget no nos la indica). Para utilizar la memoria compartida debemos antes vincularla con alguna variable de nuestro código. De esta manera, siempre que usemos la variable vinculada estaremos utilizando la variable compartida. Para establecer un vínculo utilizamos la función shmat.

Sintaxis:

```
void *shmat(int shmid, char *shmaddr, int shmflag);
```

Parámetros:

shmid: Es el identificador de la memoria compartida. Lo habremos obtenido con la llamada shmget

shmaddr: Normalmente, valdrá 0 o NULL que indicará al sistema operativo que busque esa zona de memoria en una zona de memoria libre, no en una dirección absoluta

shmflg: permisos. Por ejemplo. Aunque hayamos obtenido una zona de memoria con permiso para escritura o lectura, podemos vincularla a una variable para solo lectura. Si no queremos cambiar los permisos usamos 0

Retorno:

Se retorna un puntero a la zona de memoria compartida

Se retorna -1 en caso de algún error

2.2 Ejercicio 2

Capture, compile y ejecute el siguiente programa. Observe su funcionamiento.

Codigo capturado

```
programa2.c x
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <string.h>
4  #define VALOR 1
5
6  int main(void) {
7      int desc_arch[2];
8      char bufer[100];
9      if(pipe(desc_arch) != 0) exit(1);
10
11     if(fork() == 0) {
12         while(VALOR) {
13             read(desc_arch[0], bufer, sizeof(bufer));
14             printf("Se recibió: %s\n", bufer);
15         }
16     }
17
18     while(VALOR) {
19         gets(bufer);
20         write(desc_arch[1], bufer, strlen(bufer)+1);
21     }
22 }
```

Compilación y ejecución del programa

```
Linux Lite Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
jg ~ > Desktop > Practica5 gcc programa2.c -o programa2
programa2.c: In function 'main':
programa2.c:21:3: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
    gets(bufer);
    ^~~~
    fgets
/tmp/ccuctt5k.o: In function 'main':
programa2.c:(.text+0x77): warning: the 'gets' function is dangerous and should not be used.
jg ~ > Desktop > Practica5 ./programa2
Hola
Se recibió: Hola
Prueba
Se recibió: Prueba
Del
Se recibió: Del
Programa numero 2
Se recibió: Programa numero 2
█
```

2.3 Ejercicio 3

Capture, compile y ejecute los siguientes programas. Observe su funcionamiento. Ejecute de la siguiente forma: C:\>nombre_programa_padre nombre_programa_hijo

Código capturado

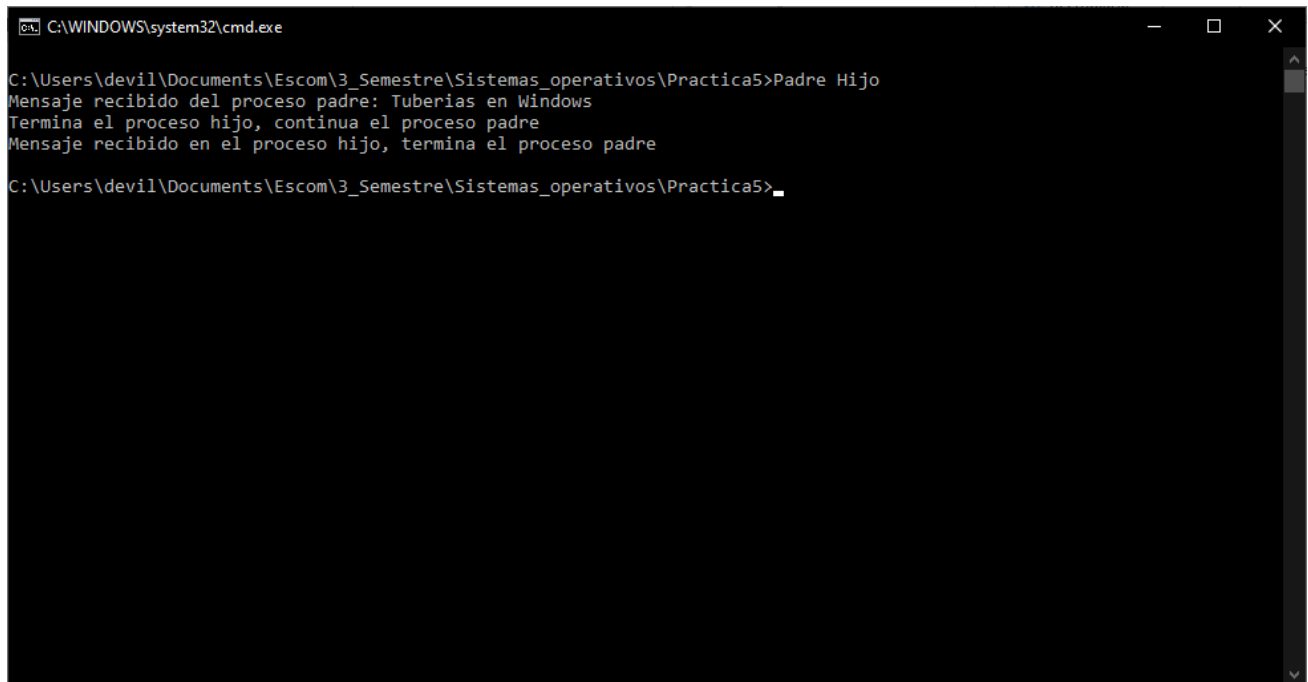
Código de padre

```
1  #include "windows.h"
2  #include "stdio.h"
3  #include "string.h"
4  int main (int argc, char *argv[])
5  {
6      char mensaje[]="Tuberias en Windows";
7      DWORD escritos;
8      HANDLE hLecturaPipe, hEscrituraPipe;
9      PROCESS_INFORMATION piHijo;
10     STARTUPINFO siHijo;
11     SECURITY_ATTRIBUTES pipeSeg =
12     {sizeof(SEcurity_ATTRIBUTES), NULL, TRUE};
13     /* Obtención de información para la inicialización del proceso hijo */
14     GetStartupInfo (&siHijo);
15     /* Creación de la tubería sin nombre */
16     CreatePipe (&hLecturaPipe, &hEscrituraPipe, &pipeSeg, 0);
17     /* Escritura en la tubería sin nombre */
18     WriteFile(hEscrituraPipe, mensaje, strlen(mensaje)+1, &escritos, NULL);
19
20     siHijo.hStdInput = hLecturaPipe;
21     siHijo.hStdError = GetStdHandle (STD_ERROR_HANDLE);
22     siHijo.hStdOutput = GetStdHandle (STD_OUTPUT_HANDLE);
23     siHijo.dwFlags = STARTF_USESTDHANDLES;
24     CreateProcess (NULL, argv[1], NULL, NULL,
25     TRUE, /* Hereda el proceso hijo los manejadores de la tubería del padre */
26     0, NULL, NULL, &siHijo, &piHijo);
27     WaitForSingleObject (piHijo.hProcess, INFINITE);
28     printf("Mensaje recibido en el proceso hijo, termina el proceso padre\n");
29     CloseHandle(hLecturaPipe);
30     CloseHandle(hEscrituraPipe);
31     CloseHandle(piHijo.hThread);
32     CloseHandle(piHijo.hProcess);
33     return 0;
34 }
```

Código Hijo

```
1  /* Programa hijo.c */
2  #include "windows.h"
3  #include "stdio.h"
4  int main ()
5  {
6      char mensaje[20];
7      DWORD leidos;
8      HANDLE hStdIn = GetStdHandle(STD_INPUT_HANDLE);
9      SECURITY_ATTRIBUTES pipeSeg =
10     {sizeof(SEcurity_ATTRIBUTES), NULL, TRUE};
11     /* Lectura desde la tubería sin nombre */
12     ReadFile(hStdIn, mensaje, sizeof(mensaje), &leidos, NULL);
13     printf("Mensaje recibido del proceso padre: %s\n", mensaje);
14     CloseHandle(hStdIn);
15     printf("Termina el proceso hijo, continua el proceso padre\n");
16     return 0;
17 }
```


Compilación y ejecución del programa



```
C:\WINDOWS\system32\cmd.exe

C:\Users\devil\Documents\Escom\3_Semestre\Sistemas_operativos\Practica5>Padre Hijo
Mensaje recibido del proceso padre: Tuberias en Windows
Termina el proceso hijo, continua el proceso padre
Mensaje recibido en el proceso hijo, termina el proceso padre

C:\Users\devil\Documents\Escom\3_Semestre\Sistemas_operativos\Practica5>_
```

2.4 Ejercicio 4

Programa una aplicación que cree un proceso hijo a partir de un proceso padre, el proceso padre enviará al proceso hijo, a través de una tubería, dos matrices de 10 x 10 a multiplicar por parte del hijo, mientras tanto el proceso hijo creará un hijo de él, al cual enviará dos matrices de 10 x 10 a sumar en el proceso hijo creado, nuevamente el envío de estos valores será a través de una tubería. Una vez calculado el resultado de la suma, el proceso hijo del hijo devolverá la matriz resultante a su abuelo (vía tubería). A su vez, el proceso hijo devolverá la matriz resultante de la multiplicación que realizó a su padre. Finalmente, el proceso padre obtendrá la matriz inversa de cada una de las matrices recibidas y el resultado lo guardará en un archivo para cada matriz inversa obtenida. Programe esta aplicación tanto para Linux como para Windows utilizando las tuberías de cada sistema operativo

2.4.1 Codificación en Linux

Codigo fuente

```
programa4.c
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <string.h>
4  #include <stdlib.h>
5  #include <math.h>
6
7
8  float determinant(float a[10][10], float k) {
9      float s = 1, det = 0, b[10][10];
10     int i, j, m, n, c;
11     if (k == 1)
12     {
13         return (a[0][0]);
14     }
15     else
16     {
17         det = 0;
18         for (c = 0; c < k; c++)
19         {
20             m = 0;
21             n = 0;
22             for (i = 0; i < k; i++)
23             {
24                 for (j = 0; j < k; j++)
25                 {
26                     b[i][j] = 0;
27                     if (i != 0 && j != c)
28                     {
29                         b[m][n] = a[i][j];
30                         if (n < (k - 2))
31                             n++;
32                         else
33                         {
34                             n = 0;
35                             m++;
36                         }
37                     }
38                 }
39             }
40             det = det + s * (a[0][c] * determinant(b, k - 1));
41             s = -1 * s;
42         }
43     }
44     return (det);
45 }
46
47
```

```

47
48 void transpose(float num[10][10], float fac[10][10], float r, FILE* archivo){
49     int i, j;
50     r = 10;
51
52     float b[10][10], inverse[10][10], d;
53
54     for (i = 0; i < r; i++)
55     {
56         for (j = 0; j < r; j++)
57         {
58             b[i][j] = fac[j][i];
59         }
60     }
61     d = determinant(num, r);
62     for (i = 0; i < r; i++)
63     {
64         for (j = 0; j < r; j++)
65         {
66             inverse[i][j] = b[i][j] / d;
67         }
68     }
69
70     fprintf(archivo, "The inverse of matrix is : \n");
71
72     for (i = 0; i < r; i++) {
73         for (j = 0; j < r; j++) {
74             fprintf(archivo, "\t%f", inverse[i][j]);
75         }
76         fprintf(archivo, "\n");
77     }
78 }
79
80
81 void cofactor(float num[10][10], float f, FILE* archivo){
82     float b[10][10], fac[10][10];
83     int p, q, m, n, i, j;
84     for (q = 0; q < f; q++)
85     {
86         for (p = 0; p < f; p++)
87         {
88             m = 0;
89             n = 0;
90             for (i = 0; i < f; i++)
91             {
92                 for (j = 0; j < f; j++)
93                 {
94                     if (i != q && j != p)
95                     {
96                         b[m][n] = num[i][j];
97                         if (n < (f - 2))
98                             n++;
99                     }
100                     else
101                     {
102                         n = 0;
103                         m++;
104                     }
105                 }
106             }
107             fac[q][p] = pow(-1, q + p) * determinant(b, f - 1);
108         }
109     }
110     transpose(num, fac, f, archivo);
111 }
112
113
114 void conversionMat(int origi[][10], float transf[][10]){
115
116     for(int i=0; i<10; i++){
117
118         for(int j=0; j<10; j++){
119
120             transf[i][j]= (float) origi[i][j];
121         }
122     }
123 }
124
125
126

```

```

120
127 void sumMatriz(int matrices[][10][10], int result[][10]){
128
129     for(int i=0; i<10; i++){
130         for(int j=0; j<10; j++){
131             result[i][j]= matrices[0][i][j] +matrices[1][i][j];
132         }
133     }
134 }
135
136
137 void multMatriz(int matrices[][10][10], int result[][10]){
138
139     for(int k=0; k<10; k++){
140         for(int i=0; i <10; i++){
141
142             int elemento=0;
143
144             for(int j=0; j<10 ; j++){
145                 elemento += (matrices[0][k][j] * matrices[1][j][i]);
146             }
147             result[k][i] = elemento;
148         }
149     }
150 }
151
152
153 void llenaMatriz(int arreglo[][10][10], int numMatrices){
154
155     for(int i=0; i<numMatrices; i++){
156
157         printf("Introduzca los valores de la matriz: %d \n", (i+1));
158
159         for(int j=0; j<10; j++){
160             for(int k=0; k<10; k++){
161                 scanf("%d", &arreglo[i][j][k]);
162             }
163             printf("\n");
164         }
165         printf("\n");
166     }
167 }
168
169
170
171 void imprimeMatriz(int arreglo[][10][10], int numMatrices){
172
173     for(int i=0; i<numMatrices; i++){
174
175         printf("Matriz %d :\n", (i+1));
176
177         for(int j=0; j<10; j++){
178             for(int k=0; k<10; k++){
179                 printf("%d ", arreglo[i][j][k]);
180             }
181             printf("\n");
182         }
183         printf("\n");
184     }
185 }
186
187 }
188
189
190 void imprimeMatrizuno(int arreglo[][10]){
191
192     for(int j=0; j<10; j++){
193         for(int k=0; k<10; k++){
194             printf("%d ", arreglo[j][k]);
195         }
196         printf("\n");
197     }
198     printf("\n");
199 }
200
201
202 void imprimeMatrizflo(float arreglo[][10]){
203
204     for(int j=0; j<10; j++){
205         for(int k=0; k<10; k++){
206             printf("%f ", arreglo[j][k]);
207         }
208         printf("\n");
209     }
210     printf("\n");
211 }
212
213

```

```

214 int main(){
215     //Identificadores de procesos
216     pid_t pidHijo, pidNieto;
217
218     int matrices1[2][10][10];
219
220     printf("Ingrese las matrices que se multiplicaran \n");
221     llenaMatriz(matrices1, 2);
222
223
224     //Arreglos que contendran los descriptors de archivos (file descriptors)
225     int fds1[2], fds2[2], fds3[2], fds4[2];
226
227     //Buffers
228     int buff[2][10][10];
229
230     int buff3[10][10];
231     int buff4[10][10];
232
233
234     if(pipe(fds1) != 0) exit(1);
235
236     if(pipe(fds3) != 0) exit(1);
237
238     if(pipe(fds4) != 0) exit(1);
239
240
241     if((pidHijo=fork())==0){
242         //Estamos dentro del proceso hijo
243
244         //Recibimos las 2 matrices a multiplicar desde el padre
245         close(fds1[1]);
246         read(fds1[0],buff,sizeof(buff));
247
248         //Realizamos la multiplicacion de matrices y devolvemos la multiplicacion en la variable "multResul"
249         int multResul[10][10];
250         multMatriz(buff, multResul);
251
252
253         //Le enviaremos al proceso nieto dos matrices para que el haga una suma
254         //creamos y llenamos esas dos matrices
255         int matrices2[2][10][10];
256
257         int buff2[2][10][10];
258
259         if(pipe(fds2) != 0) exit(1);
260
261         printf("Ingrese las matrices que se sumaran \n");
262         llenaMatriz(matrices2, 2);
263
264
265         //el hijo va a crear un proceso hijo (nieto)
266         if((pidNieto=fork())==0){
267             //Estamos dentro del proceso nieto
268
269             //Recibimos las dos matrices desde el proceso hijo
270             close(fds2[1]);
271             read(fds2[0],buff2,sizeof(buff2));
272
273             int sumResul[10][10];
274             sumMatriz(buff2, sumResul);
275
276             //estamos en el nieto y enviamos la respuesta al padre
277             close(fds4[0]);
278             write(fds4[1],sumResul,sizeof(sumResul));
279
280         }else{
281             //Estamos dentro del proceso hijo
282
283             //Enviamos las 2 matrices del proceso hijo al proceso nieto
284             close(fds2[0]);
285             write(fds2[1],matrices2,sizeof(matrices2));
286
287             //Estanos dentro del hijo y enviamos la matriz resultante de la multiplicacion al padre
288             close(fds3[0]);
289             write(fds3[1],multResul,sizeof(multResul));
290
291             wait(0);
292         }
293
294         //Estanos dentro del hijo y enviamos la matriz resultante de la multiplicacion al padre
295         close(fds3[0]);
296         write(fds3[1],multResul,sizeof(multResul));
297
298     }else{
299         //Estamos dentro del proceso padre
300
301         //Enviamos al proceso hijo las dos matrices
302         close(fds1[0]);
303         write(fds1[1],matrices1,sizeof(matrices1));
304
305         //Recibimos del proceso hijo la matriz resultante de la multiplicacion
306         close(fds3[1]);
307         read(fds3[0],buff3, sizeof(buff3));
308         printf("Del hijo nos llevo la sig matriz (Multiplicacion de matrices): \n");
309         imprimeMatrizuno(buff3);
310
311     }
312 }

```

```

315
316 //Recibimos del proceso nieto la matriz resultante de la suma
317 close(fds4[1]);
318 read(fds4[0],buff4, sizeof(buff4));
319 printf("Del nieto nos llego la sig matriz (Suma de matrices): \n");
320 imprimeMatrizuno(buff4);
321
322 wait(0); //Esperamos a que el hijo complete sus cosas
323
324
325
326 //Convertimos matriz de int a float
327 float inversaUno[10][10];
328 conversionMat(buff3, inversaUno);
329 //Guardamos en un archivo la matriz inversa resultante de la multiplicacion en caso de que exista
330 FILE* archivo = fopen("salida1.txt", "w");
331 float d1 = determinant(inversaUno, 10);
332
333 if(d1 != 0)
334     cofactor(inversaUno, 10, archivo);
335 else
336     fprintf(archivo, "La matriz no tiene inversa\n");
337
338 fclose(archivo);
339
340
341
342 //Convertimos matriz de int a float
343 float inversaDos[10][10];
344 conversionMat(buff4, inversaDos);
345
346 //Guardamos en un archivo la matriz inversa resultante de la suma en caso de que exista
347 FILE* archivo2 = fopen("salida2.txt", "w");
348 float d2 = determinant(inversaDos, 10);
349
350 if(d2 != 0)
351     cofactor(inversaDos, 10, archivo2);
352 else
353     fprintf(archivo2, "La matriz no tiene inversa\n");
354
355 fclose(archivo2);
356
357 }
358
359 return 0;
360 }

```

Compilación y ejecución del programa

```

Linux Lite Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
jg ~ > Desktop > Practica5 gcc -o programa4 programa4.c -lm
programa4.c: In function 'main':
programa4.c:296:5: warning: implicit declaration of function 'wait'; did you mean 'main'? [-Wimplicit-function-decl
ration]
    wait(0);
    ^~~~
main
jg ~ > Desktop > Practica5 ./programa4
Ingrese las matrices que se multiplicaran
Introduzca los valores de la matriz: 1
1 2 1 1 1 2 2 2 1 2
1 2 1 2 2 2 1 1 2 2
1 1 2 2 1 2 2 2 2 2
2 1 2 2 1 1 1 2 1 1
1 2 1 1 1 2 1 2 1 1
2 2 2 2 2 1 1 1 1 1
1 1 1 2 2 2 1 1 1 2
2 2 2 1 1 1 2 2 2 1
2 1 2 1 2 1 2 1 2 3
3 2 2 2 1 1 3 3 1 2

```

```
Linux Lite Terminal
Archivo Editar Ver Terminal Pestañas Ayuda

Introduzca los valores de la matriz: 2
2 3 1 2 3 2 1 2 3 1
1 1 1 2 2 2 3 3 3 2
2 2 2 3 3 3 1 1 2 1
1 2 2 2 1 1 2 3 1 3
2 3 2 1 3 2 1 3 2 2
3 2 1 3 2 1 3 2 2 2
3 3 3 3 2 2 2 1 1 1
2 2 3 3 1 1 2 1 2 3
1 2 3 2 1 3 2 1 2 3
3 3 3 2 2 1 2 1 3 2
```

```
Linux Lite Terminal
Archivo Editar Ver Terminal Pestañas Ayuda

Ingrese las matrices que se sumaran
Introduzca los valores de la matriz: 1
1 1 1 2 1 2 1 2 3 2
1 1 1 1 2 2 3 5 6 6
1 3 2 5 7 7 5 6 5 2
3 6 4 7 8 9 2 1 6 3
3 3 3 2 2 2 1 1 2 2
2 2 2 3 3 5 4 7 8 6
2 3 6 4 6 8 7 9 1 3
2 5 8 9 4 2 5 8 4 1
1 1 2 2 3 6 9 8 5 6
3 6 8 7 4 1 5 6 8 7

Introduzca los valores de la matriz: 2
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
```

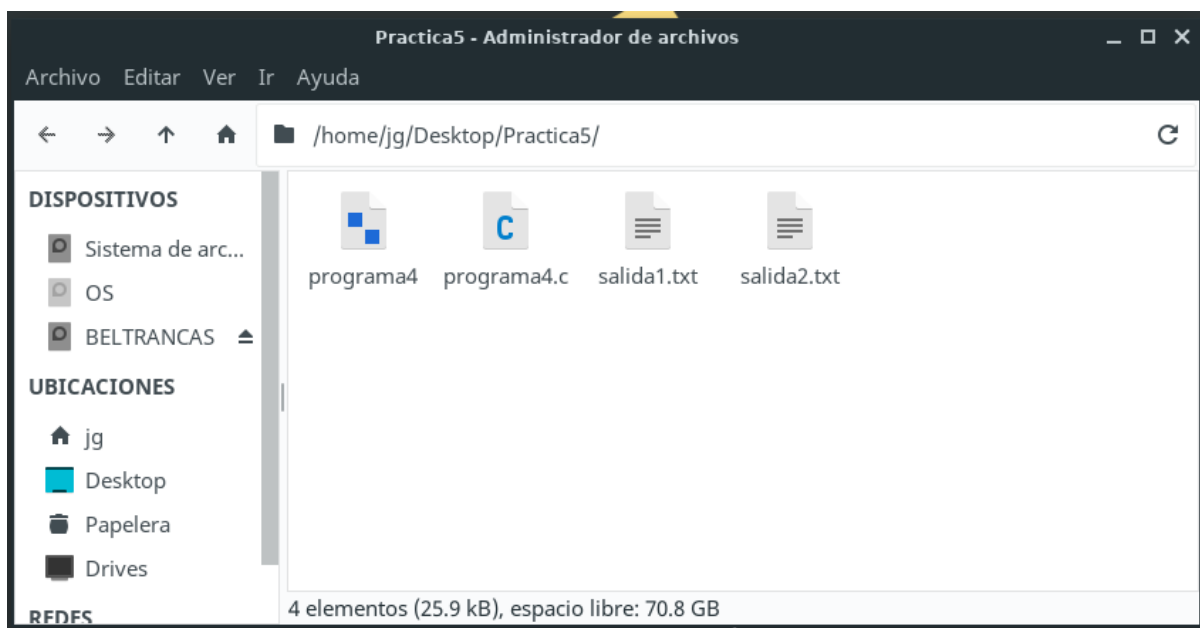
```
Linux Lite Terminal
Archivo Editar Ver Terminal Pestañas Ayuda
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1

Del hijo nos llego la sig matriz (Multiplicacion de matrices):
32 34 32 36 29 25 31 26 32 30
31 36 33 35 31 28 32 31 34 34
35 39 38 41 32 30 33 28 34 35
27 32 29 33 28 25 25 25 29 28
26 28 26 31 25 22 27 24 28 27
28 34 29 33 32 28 27 30 32 29
29 33 29 31 28 23 27 27 29 29
31 36 34 38 32 31 30 27 34 31
36 42 38 38 36 32 30 28 37 32
41 47 43 48 40 35 37 34 42 38

Del nieto nos llego la sig matriz (Suma de matrices):
2 2 2 3 2 3 2 3 4 3
2 2 2 3 3 4 6 7 7
2 4 3 6 8 8 6 7 6 3
4 7 5 8 9 10 3 2 7 4
4 4 4 3 3 3 2 2 3 3
3 3 3 4 4 6 5 8 9 7
3 4 7 5 7 9 8 10 2 4
3 6 9 10 5 3 6 9 5 2
2 2 3 3 4 7 10 9 6 7
4 7 9 8 5 2 6 7 9 8

jg ~ > Desktop > Practica5
```

Observamos que dentro de la carpeta donde se encuentra el archivo llamado programa4.c se generaron los archivos de salida que contiene las matrices inversas, en caso de existir



Observamos que el contenido de dichos archivos es el siguiente

```
salida1.txt
Archivo  Editar  Buscar  Opciones  Ayuda
1 La matriz no tiene inversa
2
```

```
salida2.txt
Archivo  Editar  Buscar  Opciones  Ayuda
1 The inverse of matrix is :
2 2.561458 -1.366809 1.088406 -0.998837 0.120202 -0.192323 0.276852 -1.058162 -0.376890 0.905777
3 -7.178678 4.819771 -3.047471 2.686719 1.167285 -0.077180 -1.227608 3.568825 1.265034 -3.481360
4 2.801510 -3.011208 1.410735 -1.302240 -0.628261 0.700456 0.733618 -1.935634 -0.693384 2.052844
5 0.797762 0.669418 -0.178253 0.159309 -0.157148 -0.719991 -0.084529 0.320279 0.098266 -0.332576
6 2.211544 -1.393730 1.295740 -0.960394 -0.288373 -0.167174 0.426134 -1.265995 -0.546502 1.220517
7 -0.552854 0.109806 -0.396444 0.313706 -0.044037 0.260935 -0.004653 0.219465 0.121180 -0.267324
8 0.506952 -0.695840 0.400266 -0.306763 0.015399 0.003933 -0.009694 -0.376115 0.105633 0.419266
9 -1.765247 1.490279 -0.786241 0.624993 0.296904 -0.046474 -0.251648 1.036060 0.216529 -1.053011
10 1.345372 -2.000499 0.985321 -0.814103 -0.266825 0.741206 0.243505 -1.151997 -0.373511 1.228051
11 -0.542015 1.496372 -0.717942 0.575140 0.002548 -0.661220 -0.116712 0.671578 0.226112 -0.729186
12
```

Para verificar las marices inversas se utilizo una calculadora de matrices online

Primero observaremos que la primera matriz no posee inversa, por lo que en el documento se imprime la leyenda “La matriz no tiene inversa”

Matriz A:

32	34	32	36	29	25	31	26	32	30
31	36	33	35	31	28	32	31	34	34
35	39	38	41	32	30	33	28	34	35
27	32	29	33	28	25	25	25	29	28
26	28	26	31	25	22	27	24	28	27
28	34	29	33	32	28	27	30	32	29
29	33	29	31	28	23	27	27	29	29
31	36	34	38	32	31	30	27	34	31
36	42	38	38	36	32	30	28	37	32
41	47	43	48	40	35	37	34	42	38

Celdas Limpiar + -

Determinante	Matriz Inversa	Matriz Transpuesta
Rango	Multiplicar por 2	Matriz Triangular
Matriz Diagonal	Matriz elevada a 2	Factorización LU
Factorización de Cho...		

Debido a que el determinante es 0 la matriz no tiene inversa

$$\begin{vmatrix} 32 & 34 & 32 & 36 & 29 & 25 & 31 & 26 & 32 & 30 \\ 31 & 36 & 33 & 35 & 31 & 28 & 32 & 31 & 34 & 34 \\ 35 & 39 & 38 & 41 & 32 & 30 & 33 & 28 & 34 & 35 \\ 27 & 32 & 29 & 33 & 28 & 25 & 25 & 25 & 29 & 28 \\ 26 & 28 & 26 & 31 & 25 & 22 & 27 & 24 & 28 & 27 \\ 28 & 34 & 29 & 33 & 32 & 28 & 27 & 30 & 32 & 29 \\ 29 & 33 & 29 & 31 & 28 & 23 & 27 & 27 & 29 & 29 \\ 31 & 36 & 34 & 38 & 32 & 31 & 30 & 27 & 34 & 31 \\ 36 & 42 & 38 & 38 & 36 & 32 & 30 & 28 & 37 & 32 \\ 41 & 47 & 43 & 48 & 40 & 35 & 37 & 34 & 42 & 38 \end{vmatrix} = 0$$

Matriz A:

32	34	32	36	29	25	31	26	32	30
31	36	33	35	31	28	32	31	34	34
35	39	38	41	32	30	33	28	34	35
27	32	29	33	28	25	25	25	29	28
26	28	26	31	25	22	27	24	28	27
28	34	29	33	32	28	27	30	32	29
29	33	29	31	28	23	27	27	29	29
31	36	34	38	32	31	30	27	34	31
36	42	38	38	36	32	30	28	37	32
41	47	43	48	40	35	37	34	42	38

El determinante de la matriz es cero, la matriz es no invertible.

OK

Para la segunda matriz observamos que la matriz resultante por la calculadora es igual a la mostrada en el archivo llamado “salida2.txt”

Matriz B:

$$\begin{pmatrix} 2 & 2 & 2 & 3 & 2 & 3 & 2 & 3 & 4 & 3 \\ 2 & 2 & 2 & 2 & 3 & 3 & 4 & 6 & 7 & 7 \\ 2 & 4 & 3 & 6 & 8 & 8 & 6 & 7 & 6 & 3 \\ 4 & 7 & 5 & 8 & 9 & 10 & 3 & 2 & 7 & 4 \\ 4 & 4 & 4 & 3 & 3 & 3 & 2 & 2 & 3 & 3 \\ 3 & 3 & 3 & 4 & 4 & 6 & 5 & 8 & 9 & 7 \\ 3 & 4 & 7 & 5 & 7 & 9 & 8 & 10 & 2 & 4 \\ 3 & 6 & 9 & 10 & 5 & 3 & 6 & 9 & 5 & 2 \\ 2 & 2 & 3 & 3 & 4 & 7 & 10 & 9 & 6 & 7 \\ 4 & 7 & 9 & 8 & 5 & 2 & 6 & 7 & 9 & 8 \end{pmatrix}$$

Celdas Limpiar + -

Determinante	Matriz Inversa	Matriz Transpuesta
Rango	Multiplicar por 2	Matriz Triangular
Matriz Diagonal	Matriz elevada a 2	Factorización LU
Factorización de Cho...		

$$\begin{pmatrix} 2 & 2 & 2 & 3 & 2 & 3 & 2 & 3 & 4 & 3 \\ 2 & 2 & 2 & 2 & 3 & 3 & 4 & 6 & 7 & 7 \\ 2 & 4 & 3 & 6 & 8 & 8 & 6 & 7 & 6 & 3 \\ 4 & 7 & 5 & 8 & 9 & 10 & 3 & 2 & 7 & 4 \\ 4 & 4 & 4 & 3 & 3 & 3 & 2 & 2 & 3 & 3 \\ 3 & 3 & 3 & 4 & 4 & 6 & 5 & 8 & 9 & 7 \\ 3 & 4 & 7 & 5 & 7 & 9 & 8 & 10 & 2 & 4 \\ 3 & 6 & 9 & 10 & 5 & 3 & 6 & 9 & 5 & 2 \\ 2 & 2 & 3 & 3 & 4 & 7 & 10 & 9 & 6 & 7 \\ 4 & 7 & 9 & 8 & 5 & 2 & 6 & 7 & 9 & 8 \end{pmatrix}^{(-1)} = \begin{pmatrix} 2.56 & -1.37 & 1.09 & -0.999 & 0.120 & -0.192 & 0.277 & -1.06 & -0.377 & 0.906 \\ -7.18 & 4.82 & -3.05 & 2.69 & 1.17 & -0.0772 & -1.23 & 3.57 & 1.27 & -3.48 \\ 2.80 & -3.01 & 1.41 & -1.30 & -0.628 & 0.700 & 0.734 & -1.94 & -0.693 & 2.05 \\ 0.798 & 0.669 & -0.178 & 0.159 & -0.157 & -0.720 & -0.0845 & 0.320 & 0.0983 & -0.333 \\ 2.21 & -1.39 & 1.30 & -0.960 & -0.288 & -0.167 & 0.426 & -1.27 & -0.547 & 1.22 \\ -0.553 & 0.110 & -0.396 & 0.314 & -0.0440 & 0.261 & -0.00465 & 0.219 & 0.121 & -0.267 \\ 0.507 & -0.696 & 0.400 & -0.307 & 0.0154 & 0.00393 & -0.00969 & -0.376 & 0.1056 & 0.419 \\ -1.77 & 1.49 & -0.786 & 0.625 & 0.297 & -0.0465 & -0.252 & 1.04 & 0.217 & -1.05 \\ 1.35 & -2.00 & 0.985 & -0.814 & -0.267 & 0.741 & 0.244 & -1.15 & -0.374 & 1.23 \\ -0.542 & 1.50 & -0.718 & 0.575 & 0.00255 & -0.661 & -0.117 & 0.672 & 0.226 & -0.729 \end{pmatrix}$$

Insertar en A
Insertar en B
Limpiar

2.4.2 Codificación en Windows

Código fuente

Código de padre

```
1  #include <windows.h>
2  #include <stdio.h>
3
4  float determinant(float a[25][25], float k) {
5      float s = 1, det = 0, b[25][25];
6      int i, j, m, n, c;
7      if (k == 1)
8      {
9          return (a[0][0]);
10     }
11     else
12     {
13         det = 0;
14         for (c = 0; c < k; c++)
15         {
16             m = 0;
17             n = 0;
18             for (i = 0; i < k; i++)
19             {
20                 for (j = 0; j < k; j++)
21                 {
22                     b[i][j] = 0;
23                     if (i != 0 && j != c)
24                     {
25                         b[m][n] = a[i][j];
26                         if (n < (k - 2))
27                             n++;
28                         else
29                         {
30                             n = 0;
31                             m++;
32                         }
33                     }
34                 }
35             }
36             det = det + s * (a[0][c] * determinant(b, k - 1));
37             s = -1 * s;
38         }
39     }
40
41     return (det);
42 }
43
44 void cofactor(float num[25][25], float f, FILE* archivo){
45     float b[25][25], fac[25][25];
46     int p, q, m, n, i, j;
47     for (q = 0; q < f; q++)
48     {
49         for (p = 0; p < f; p++)
50         {
51             m = 0;
52             n = 0;
53             for (i = 0; i < f; i++)
54             {
55                 for (j = 0; j < f; j++)
56                 {
57                     if (i != q && j != p)
58                     {
59                         b[m][n] = num[i][j];
60                         if (n < (f - 2))
61                             n++;
62                         else
63                         {
64                             n = 0;
65                             m++;
66                         }
67                     }
68                 }
69             }
70             fac[q][p] = pow(-1, q + p) * determinant(b, f - 1);
```

```

71 | }
72 | }
73 | transpose(num, fac, f, archivo);
74 | }
75 |
76 | void transpose(float num[25][25], float fac[25][25], float r, FILE* archivo){
77 |     int i, j;
78 |     r = 10;
79 |
80 |     float b[25][25], inverse[25][25], d;
81 |
82 |     for (i = 0; i < r; i++)
83 |     {
84 |         for (j = 0; j < r; j++)
85 |         {
86 |             b[i][j] = fac[j][i];
87 |         }
88 |     }
89 |     d = determinant(num, r);
90 |     for (i = 0; i < r; i++)
91 |     {
92 |         for (j = 0; j < r; j++)
93 |         {
94 |             inverse[i][j] = b[i][j] / d;
95 |         }
96 |     }
97 |
98 |     fprintf(archivo, "The inverse of matrix is : \n");
99 |
100 |    for (i = 0; i < r; i++) {
101 |        for (j = 0; j < r; j++) {
102 |            fprintf(archivo, "\t%f", inverse[i][j]);
103 |        }
104 |        fprintf(archivo, "\n");
105 |    }
106 | }
107 |
108 |
109 | int main(void){
110 |     //Create pipe
111 |     return 0;
112 |     SECURITY_ATTRIBUTES saAttr;
113 |     saAttr.nLength = sizeof(SECURITY_ATTRIBUTES);
114 |     saAttr.bInheritHandle = TRUE;
115 |     saAttr.lpSecurityDescriptor = NULL;
116 |
117 |     HANDLE hRead1, hWrite1, hRead2, hWrite2;
118 |
119 |     CreatePipe(&hRead1, &hWrite1, &saAttr, 0);
120 |     CreatePipe(&hRead2, &hWrite2, &saAttr, 0);
121 |     ///////////////
122 |
123 |     //Send Menssage
124 |     int i, j;
125 |     float szBuffer[10][20];
126 |     for(i = 0; i < 10; i++)
127 |         for(j = 0; j < 20; j++)
128 |             szBuffer[i][j] = rand() % 10;
129 |     DWORD dwBufferSize = sizeof(szBuffer);
130 |     DWORD dwNoBytesWrite;
131 |
132 |     BOOL bWriteFile = WriteFile(hWrite1, szBuffer, dwBufferSize, &dwNoBytesWrite, NULL);
133 |     ///////////////
134 |
135 |     //Create child
136 |     STARTUPINFO si1;
137 |     GetStartupInfo(&si1);
138 |     PROCESS_INFORMATION pi1;
139 |     ZeroMemory( &si1, sizeof(si1) );
140 |     ZeroMemory( &pi1, sizeof(pi1) );

```

```

141     si1.cb = sizeof(STARTUPINFO);
142     si1.hStdError = GetStdHandle (STD_ERROR_HANDLE);
143     si1.hStdOutput = hWrite2;
144     si1.hStdInput = hRead1;
145     si1.dwFlags = STARTF_USESTDHANDLES;
146
147     printf("Padre: %f\n", szBuffer[0][0]);
148     BOOL process1 = CreateProcess("son.exe", NULL, NULL, NULL, TRUE, 0, NULL, NULL, &si1, &pi1);
149     if(process1 == FALSE) printf("No");
150
151     WaitForSingleObject(pi1.hProcess, INFINITE);
152
153     CloseHandle(pi1.hProcess);
154     CloseHandle(pi1.hThread);
155
156     ///////////////////////////////////
157
158
159     //Receive message
160     DWORD dwNoBytesRead;
161     printf("Padre: %f\n", szBuffer[0][0]);
162     BOOL bReadFile = ReadFile(hRead2, szBuffer, dwBufferSize, &dwNoBytesRead, NULL);
163     printf("Padre: %f\n", szBuffer[0][0]);
164
165     CloseHandle(hRead1);
166     CloseHandle(hWrite1);
167     CloseHandle(hRead2);
168     CloseHandle(hWrite2);
169     ///////////////////////////////////
170
171
172     //Write Results
173     float a[10][10], b[10][10];
174
175     for(i = 0; i < 10; i++){

```

```

176         for(j = 0; j < 20; j++){
177             if(j < 10)
178                 a[i][j] = szBuffer[i][j];
179             else
180                 b[i][j - 10] = szBuffer[i][j];
181         }
182     }
183     FILE* archivo = fopen("salida1.txt", "w");
184
185     float d1 = determinant(a, 10);
186
187     if(d1 != 0)
188         cofactor(a, 10, archivo);
189     else
190         fprintf(archivo, "La matriz no tiene inversa\n");
191
192     fclose(archivo);
193
194     archivo = fopen("salida2.txt", "w");
195     float d2 = determinant(b, 10);
196
197     if(d2 != 0)
198         cofactor(b, 10, archivo);
199     else
200         fprintf(archivo, "La matriz no tiene inversa\n");
201
202     fclose(archivo);
203     ///////////////////////////////////
204     return 0;
205

```

Código Hijo

```
1  #include <windows.h>
2  #include <stdio.h>
3
4  int main(void){
5      //Recibe data
6      HANDLE hStdin = GetStdHandle(STD_INPUT_HANDLE);
7      HANDLE hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
8
9      DWORD dwBufferSize = sizeof(szBuffer);
10     DWORD dwNoBytesRead;
11     DWORD dwNoBytesWrite;
12     int szBuffer[10][20];
13     BOOL bWriteFile;
14     BOOL bReadFile;
15
16     printf("Hijo :%f \n", szBuffer[0][0]);
17     bReadFile = ReadFile(hStdin, szBuffer, dwBufferSize, &dwNoBytesRead, NULL);
18     CloseHandle(hStdin);
19
20     ////////////
21
22     //Prepare data
23     int i, j;
24     float a[10][10], b[10][10];
25
26     for(i = 0; i < 10; i++){
27         for(j = 0; j < 20; j++){
28             if(j < 10)
29                 a[i][j] = szBuffer[i][j];
30             else
31                 b[i][j - 10] = szBuffer[i][j];
32         }
33     }
34     ////////////
35
36     //Send data to child
37     SECURITY_ATTRIBUTES saAttr;
38     saAttr.nLength = sizeof(SECURITY_ATTRIBUTES);
39     saAttr.bInheritHandle = TRUE;
40     saAttr.lpSecurityDescriptor = NULL;
41
42     HANDLE hRead1, hWrite1, hRead2, hWrite2;
43
44     CreatePipe(&hRead1, &hWrite1, &saAttr, 0);
45     CreatePipe(&hRead2, &hWrite2, &saAttr, 0);
46
47     bWriteFile = WriteFile(hWrite1, szBuffer, dwBufferSize, &dwNoBytesWrite, NULL);
48     ////////////
49
50     //Create child
51
52     STARTUPINFO si1;
53     GetStartupInfo(&si1);
54     PROCESS_INFORMATION pi1;
55     ZeroMemory( &si1, sizeof(si1) );
56     ZeroMemory( &pi1, sizeof(pi1) );
57     si1.cb = sizeof(STARTUPINFO);
58     si1.hStdError = GetStdHandle(STD_ERROR_HANDLE);
59     si1.hStdOutput = hWrite2;
60     si1.hStdInput = hRead1;
61     si1.dwFlags = STARTF_USESTDHANDLES;
62
63     BOOL process1 = CreateProcess("grandson.exe", NULL, NULL, NULL, TRUE, 0, NULL, NULL, &si1, &pi1);
64
65     WaitForSingleObject(pi1.hProcess, INFINITE);
66
67     CloseHandle(pi1.hProcess);
68     CloseHandle(pi1.hThread);
69
70     ////////////
```

```

71 //Receive message from child
72 bReadFile = ReadFile(hRead2, szBuffer, dwBufferSize, &dwNoBytesRead, NULL);
73
74 CloseHandle(hRead1);
75 CloseHandle(hWrite1);
76 CloseHandle(hRead2);
77 CloseHandle(hWrite2);
78 ///////////////////////////////////////////////////
79
80
81 ///Send Data to parent
82 printf("Resultado Multiplicacion:\n");
83 for(i = 0; i < 10; i++){
84     for(j = 0; j < 10; j++){
85         szBuffer[i][j] = a[i][j] * b[i][j];
86         printf("%f ", szBuffer[i][j]);
87     }
88     printf("\n");
89 }
90
91 bWriteFile = WriteFile(hStdout, szBuffer, dwBufferSize, &dwNoBytesWrite, NULL);
92 CloseHandle(hStdout);
93 ///////////////////////////////////////////////////
94
95 return 0;
96 }

```

Código Nieto

```

1 #include <windows.h>
2 #include <stdio.h>
3
4 int main(void){
5     //Recibe data
6     HANDLE hStdin = GetStdHandle(STD_INPUT_HANDLE);
7     HANDLE hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
8
9     int szBuffer[10][20];
10    DWORD dwBufferSize = sizeof(szBuffer);
11    DWORD dwNoBytesRead;
12
13    BOOL bReadFile = ReadFile(hStdin, szBuffer, dwBufferSize, &dwNoBytesRead, NULL);
14    CloseHandle(hStdin);
15
16    //Prepare Data
17    int i, j;
18    float a[10][10], b[10][10];
19
20    for(i = 0; i < 10; i++){
21        for(j = 0; j < 20; j++){
22            if(j < 10)
23                a[i][j] = szBuffer[i][j];
24            else
25                b[i][j - 10] = szBuffer[i][j];
26        }
27    }
28    ///////////////////////////////////////////////////
29
30    ///Send data to parent
31    printf("Resultado Suma:\n");
32    for(i = 0; i < 10; i++){
33        for(j = 0; j < 10; j++){
34            szBuffer[i + 10][j + 10] = a[i][j] + b[i][j];
35            printf("%f ", szBuffer[i + 10][j + 10]);
36        }
37        printf("\n");
38    }
39
40    DWORD dwNoBytesWrite;
41    BOOL bWriteFile = WriteFile(hStdout, szBuffer, dwBufferSize, &dwNoBytesWrite, NULL);
42    CloseHandle(hStdout);
43    ///////////////////////////////////////////////////
44
45    return 0;
46 }

```


Compilación y ejecución del programa

```
C:\Users\devil\Documents\Escm\3_Semestre\Sistemas_operativos\Practica5\P5_W_4_2.exe

Resultado Suma:
6.000000 12.000000 5.000000 7.000000 10.000000 5.000000 13.000000 10.000000 9.000000 10.000000
8.000000 10.000000 3.000000 11.000000 11.000000 4.000000 8.000000 15.000000 13.000000 9.000000
10.000000 5.000000 4.000000 10.000000 10.000000 13.000000 4.000000 2.000000 12.000000 16.000000
12.000000 10.000000 0.000000 11.000000 8.000000 6.000000 6.000000 3.000000 7.000000 16.000000
13.000000 12.000000 10.000000 7.000000 13.000000 8.000000 14.000000 8.000000 10.000000 9.000000
4.000000 10.000000 13.000000 10.000000 4.000000 9.000000 14.000000 9.000000 10.000000 8.000000
6.000000 0.000000 2.000000 10.000000 11.000000 7.000000 12.000000 5.000000 8.000000 9.000000
9.000000 13.000000 9.000000 3.000000 8.000000 17.000000 9.000000 9.000000 3.000000 13.000000
1.000000 15.000000 8.000000 7.000000 14.000000 4.000000 13.000000 10.000000 10.000000 4.000000
9.000000 7.000000 11.000000 12.000000 15.000000 9.000000 0.000000 15.000000 10.000000 11.000000
Resultado Multi:
5.000000 35.000000 4.000000 0.000000 9.000000 4.000000 40.000000 16.000000 14.000000 24.000000
7.000000 24.000000 20.000000 18.000000 18.000000 4.000000 7.000000 54.000000 40.000000 20.000000
21.000000 4.000000 4.000000 21.000000 21.000000 36.000000 3.000000 1.000000 27.000000 64.000000
36.000000 25.000000 0.000000 18.000000 0.000000 0.000000 0.000000 2.000000 12.000000 64.000000
36.000000 27.000000 12.000000 42.000000 0.000000 0.000000 48.000000 12.000000 9.000000 8.000000
3.000000 25.000000 36.000000 16.000000 0.000000 0.000000 49.000000 18.000000 21.000000 12.000000
0.000000 0.000000 1.000000 24.000000 0.000000 0.000000 35.000000 4.000000 7.000000 14.000000
14.000000 42.000000 18.000000 0.000000 15.000000 72.000000 0.000000 8.000000 2.000000 40.000000
0.000000 54.000000 16.000000 0.000000 48.000000 3.000000 40.000000 9.000000 16.000000 0.000000
8.000000 12.000000 24.000000 27.000000 56.000000 8.000000 0.000000 56.000000 21.000000 28.000000

-----
Process exited after 9.671 seconds with return value 0
Presione una tecla para continuar . . .
```

Los archivos generados por el programa y los cuales contendrán a las inversas de las matrices se muestran a continuación

Inversa suma

```
Inversa_suma.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
La inversa :
-0.092916 0.056365 -0.029632 0.023751 0.102433 -0.009647 -0.006355 0.007381 -0.042781 -0.017852
-0.087678 0.031571 -0.018603 0.058600 -0.004970 0.004782 -0.050284 0.025473 0.073412 -0.031124
0.103554 -0.110113 0.003621 -0.023575 0.030748 0.040145 -0.034573 -0.049002 -0.036534 0.070105
-0.233494 0.043242 -0.073988 0.120050 -0.034164 0.064894 0.060737 0.011897 0.076846 -0.001055
0.019245 -0.055527 0.002006 -0.007344 0.010311 -0.065864 0.034690 -0.007305 0.050560 0.037031
-0.198297 0.086932 0.018578 -0.008094 -0.016817 0.007338 0.037495 0.089751 0.056282 -0.054896
0.041441 -0.011685 -0.026155 -0.009233 0.012264 0.012728 0.046503 0.022282 -0.016438 -0.030697
0.054808 0.056276 -0.038737 -0.036375 -0.027057 -0.006854 0.008347 0.027732 -0.051116 0.019491
0.043069 0.057944 0.114828 -0.085648 0.023485 0.001579 -0.059118 -0.046264 -0.014922 -0.040899
0.305093 -0.128091 0.038582 -0.020062 -0.058881 -0.014367 -0.030108 -0.045455 -0.090737 0.061387
```

Inversa Multiplicación

```
Inversa_multi.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
La inversa :
-0.005336 0.009214 0.000412 0.002080 0.026561 -0.007701 -0.009767 0.000943 -0.010200 -0.008456
-0.012573 0.007667 -0.016091 0.015694 -0.007916 0.001394 0.005440 0.008441 0.015468 -0.006907
-0.011822 -0.015794 0.005078 0.000121 -0.002037 0.022188 -0.016819 -0.002592 0.004186 0.012716
-0.042305 0.017824 -0.022179 0.019757 -0.007845 0.010217 0.039390 0.013681 0.003282 -0.012308
0.000470 -0.007334 0.007390 -0.004593 0.002335 -0.009771 -0.000186 -0.004686 0.009590 0.008751
-0.002464 0.007238 0.002896 -0.010710 0.004635 -0.002407 0.005865 0.013279 -0.006764 -0.007393
0.019891 -0.002414 0.001204 -0.008588 0.006636 0.000270 0.008178 -0.000773 -0.010009 -0.003444
0.021655 0.004168 -0.010475 -0.004898 0.003843 -0.005559 -0.002439 0.003249 -0.017266 0.011461
-0.008602 0.014251 0.029398 -0.013074 0.002127 0.005322 -0.021156 -0.014223 0.013342 -0.012110
0.020747 -0.015993 0.007107 0.005372 -0.010161 0.000090 -0.003666 -0.005110 -0.003190 0.012829
```

2.5 Ejercicio 5

Capture, compile y ejecute los siguientes programas para Linux. Observe su funcionamiento.

Código capturado

Código de Cliente de memoria compartida

```
1 #include <sys/types.h>
2 #include <sys/ipc.h>
3 #include <sys/shm.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #define TAM_MEM 27
7 int main()
8 {
9     int shmid;
10    key_t llave;
11    char *shm, *s;
12    llave = 5678;
13
14    if((shmid = shmget(llave, TAM_MEM, 0666))<0){
15        perror("Error al obtener memoria compartida: shmget");
16        exit(-1);
17    }
18    if((shm = shmat(shmid, NULL, 0)) == (char*)-1){
19        perror("Error al enlazar la memoria compartida: shmat");
20        exit(-1);
21    }
22
23    for(s = shm; *s != '\0'; s++)
24        putchar(*s);
25    putchar('\n');
26    *shm = '*';
27    exit(0);
28 }
```

Código de Servidor de la Memoria Compartida

```
1 #include <sys/types.h>
2 #include <sys/ipc.h>
3 #include <sys/shm.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <unistd.h>
7 # define TAM_MEM 27
8 int main(){
9     char c;
10    int shmid;
11    key_t llave;
12    char *shm, *s;
13    llave = 5678;
14    if((shmid = shmget(llave, TAM_MEM, IPC_CREAT | 0666)) < 0){
15        perror("Error al obtener la memoria compartida: shmget");
16        exit(-1);
17    }
18    if((shm = shmat(shmid, NULL, 0 )) == (char*)-1){
19        perror("Error al enlazar la memoria compartida: shmat");
20        exit(-1);
21    }
22
23    s = shm;
24    for(c = 'a'; c<='z'; c++)
25        *s++ = c;
26    *s = '\0';
27
28    while(*shm != 'q')
29        sleep(1);
30    exit(0);
31 }
```

Compilación y ejecución del programa

Compilación y Ejecución: Cliente.

```
oliver@oliver-VirtualBox:~/Documents/practica5$ gcc inciso5.c -o  
inciso5.out  
oliver@oliver-VirtualBox:~/Documents/practica5$ ./inciso5.out  
abcdefghijklmnopqrstuvwxyz  
oliver@oliver-VirtualBox:~/Documents/practica5$
```

Compilación y Ejecución: Servidor.

```
oliver@oliver-VirtualBox:~/Documents/practica5$ gcc servidorP5.c  
-o servidorP5.out  
oliver@oliver-VirtualBox:~/Documents/practica5$ ./servidorP5.out  
oliver@oliver-VirtualBox:~/Documents/practica5$
```

2.6 Ejercicio 6

Capture, compile y ejecute los siguientes programas para Windows. Observe su funcionamiento.

Código capturado

Código del cliente de memoria compartida

```
C Servidor.c X C Cliente.c C ProcesoH.c C ProcesoN.c C MCliente.c X C MServidor.c
C: > Users > rodri > Desktop > Escuela > Sistemas > Practica5 > C MCliente.c > main(void)
1  #include <windows.h> /* Cliente de la memoria compartida */
2  #include <stdio.h>
3  #define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
4  int main(void)
5  {
6      HANDLE hArchMapeo;
7      char *idMemCompartida = "MemoriaCompatida";
8      char *apDatos, *apTrabajo, c;
9      if ((hArchMapeo = OpenFileMapping(
10         FILE_MAP_ALL_ACCESS, // acceso lectura/escritura de la memoria compartida
11         FALSE,                // no se hereda el nombre
12         idMemCompartida)      // identificador de la memoria compartida
13     ) == NULL)
14     {
15         printf("No se abrio archivo de mapeo de la memoria compartida: (%i)\n", GetLastError());
16         exit(-1);
17     }
18     if ((apDatos = (char *)MapViewOfFile(hArchMapeo, // Manejador del mapeo
19         FILE_MAP_ALL_ACCESS, // Permiso de lectura/escritura en la memoria
20         0,
21         0,
22         TAM_MEM)) == NULL)
23     {
24         printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
25         CloseHandle(hArchMapeo);
26         exit(-1);
27     }
28     for (apTrabajo = apDatos; *apTrabajo != '\0'; apTrabajo++)
29         putchar(*apTrabajo);
30     putchar('\n');
31     *apDatos = '*';
32     UnmapViewOfFile(apDatos);
33     CloseHandle(hArchMapeo);
34     exit(0);
35 }
```

Código del servidor de memoria compartida

```
C Servidor.c X C Cliente.c C ProcesoH.c C ProcesoN.c C MCliente.c C MServidor.c X
C: > Users > rodri > Desktop > Escuela > Sistemas > Practica5 > C MServidor.c > main(void)
1  #include <windows.h> /* Servidor de la memoria compartida */
2  #include <stdio.h> /* (ejecutar el servidor antes de ejecutar el cliente)*/
3  #define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
4  int main(void)
5  {
6      HANDLE hArchMapeo;
7      char *idMemCompartida = "MemoriaCompatida";
8      char *apDatos, *apTrabajo, c;
9      if ((hArchMapeo = CreateFileMapping(
10         INVALID_HANDLE_VALUE, // usa memoria compartida
11         NULL, // seguridad por default
12         PAGE_READWRITE, // acceso lectura/escritura a la memoria
13         0, // tamaño maximo parte alta de un DWORD
14         TAM_MEM, // tamaño maximo parte baja de un DWORD
15         idMemCompartida) // identificador de la memoria compartida
16     ) == NULL)
17     {
18         printf("No se mapeo la memoria compartida: (%i)\n", GetLastError());
19         exit(-1);
20     }
21     if ((apDatos = (char *)MapViewOfFile(hArchMapeo, // Manejador del mapeo
22         FILE_MAP_ALL_ACCESS, // Permiso de lectura/escritura en la memoria
23         0,
24         0,
25         TAM_MEM)) == NULL)
26     {
27         printf("No se creo la memoria compartida: (%i)\n", GetLastError());
28         CloseHandle(hArchMapeo);
29         exit(-1);
30     }
31     apTrabajo = apDatos;
32     for (c = 'a'; c <= 'z'; c++)
33         *apTrabajo++ = c;
34     *apTrabajo = '\0';
35     while (*apDatos != '*')
36         sleep(1);
37     UnmapViewOfFile(apDatos);
38     CloseHandle(hArchMapeo);
39     exit(0);
40 }
```

Compilación y ejecución del programa

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
PS C:\Users\rodri\Desktop\Escuela\Sistemas\Practica5> gcc MServidor.c -o MServidor
PS C:\Users\rodri\Desktop\Escuela\Sistemas\Practica5> ./MServidor
PS C:\Users\rodri\Desktop\Escuela\Sistemas\Practica5> █
```

1: powershell, powershell

```
PS C:\Users\rodri\Desktop\Escuela\Sistemas\Practica5> gcc MCliente.c -o MCliente
PS C:\Users\rodri\Desktop\Escuela\Sistemas\Practica5> ./MCliente
abcdefghijklmnopqrstuvwxyz
PS C:\Users\rodri\Desktop\Escuela\Sistemas\Practica5> █
```

2.7 Ejercicio 7

Programa nuevamente la aplicación del punto cuatro utilizando en esta ocasión memoria compartida en lugar de tuberías (utilice tantas memorias compartidas como requiera). Programa esta aplicación tanto para Linux como para Windows utilizando la memoria compartida de cada sistema operativo.

2.7.1 Codificación en Linux

Codigo fuente

```
1 #include <sys/types.h>
2 #include <sys/ipc.h>
3 #include <sys/shm.h>
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <unistd.h>
7 #include <wait.h>
8 #include <time.h>
9 #include <math.h>
10 #define TAM_MEM 27
11
12 void showM(double **);
13 void showMT(float transf[][10]);
14 void creaM(void **);
15 void llenaM(double **);
16
17 void conversionMat(double **matrix, float transf[][10]){
18     for(int i=0; i<10; i++){
19         for(int j=0; j<10; j++){
20             transf[i][j]= (float) matrix[i][j];
21         }
22     }
23 }
24
25 float determinant(float a[10][10], float k) {
26     float s = 1, det = 0, b[10][10];
27     int i, j, m, n, c;
28     if (k == 1)
29     {
30         return (a[0][0]);
31     }
32     else
33     {
34         det = 0;
35         for (c = 0; c < k; c++)
36         {
37             m = 0;
38             n = 0;
39             for (i = 0; i < k; i++)
40             {
41                 for (j = 0; j < k; j++)
42                 {
43                     b[i][j] = 0;
44                     if (i != 0 && j != c)
45                     {
46                         b[m][n] = a[i][j];
47                         if (n < (k - 2))
48                             n++;
49                         else
50                         {
51                             n = 0;
52                             m++;
53                         }
54                     }
55                 }
56             }
57             det = det + s * (a[0][c] * determinant(b, k - 1));
58             s = -1 * s;
59         }
60     }
61 }
62
63 }
64
```

```

65     return (det);
66 }
67
68 void transpose(float num[10][10], float fac[10][10], float r, FILE* archivo){
69     int i, j;
70     r = 10;
71
72     float b[10][10], inverse[10][10], d;
73
74     for (i = 0; i < r; i++)
75     {
76         for (j = 0; j < r; j++)
77         {
78             b[i][j] = fac[j][i];
79         }
80     }
81     d = determinant(num, r);
82     for (i = 0; i < r; i++)
83     {
84         for (j = 0; j < r; j++)
85         {
86             inverse[i][j] = b[i][j] / d;
87         }
88     }
89
90     fprintf(archivo, "The inverse of matrix is : \n");
91
92     for (i = 0; i < r; i++) {
93         for (j = 0; j < r; j++) {
94             fprintf(archivo, "\t%f", inverse[i][j]);
95         }
96         fprintf(archivo, "\n");
97     }
98 }
99
100 void cofactor(float num[10][10], float f, FILE* archivo){
101     float b[10][10], fac[10][10];
102     int p, q, m, n, i, j;
103     for (q = 0; q < f; q++)
104     {
105         for (p = 0; p < f; p++)
106         {
107             m = 0;
108             n = 0;
109             for (i = 0; i < f; i++)
110             {
111                 for (j = 0; j < f; j++)
112                 {
113                     if (i != q && j != p)
114                     {
115                         b[m][n] = num[i][j];
116                         if (n < (f - 2))
117                             n++;
118                         else
119                         {
120                             n = 0;
121                             m++;
122                         }
123                     }
124                 }
125             }
126             fac[q][p] = pow(-1, q + p) * determinant(b, f - 1);
127         }
128     }
129     transpose(num, fac, f, archivo);
130 }
131
132 int main(void){
133     srand(time(NULL));
134     int shmId1, shmId2, shmId3, shmId4, shmId5, shmIdM, flag = 0;
135     double **matrix1, **matrix2, **matrix3, **matrix4, **matrix5, **matrixM;
136     char *shm, *s, a;
137     pid_t pidi;
138
139     size_t sizeMatrix = 10 * (sizeof(void*) + 10*sizeof(double));
140     shmId1 = shmget(IPC_PRIVATE, sizeMatrix, IPC_CREAT|0600);
141     shmId2 = shmget(IPC_PRIVATE, sizeMatrix, IPC_CREAT|0600);
142     shmId3 = shmget(IPC_PRIVATE, sizeMatrix, IPC_CREAT|0600);
143     shmId4 = shmget(IPC_PRIVATE, sizeMatrix, IPC_CREAT|0600);
144     shmId5 = shmget(IPC_PRIVATE, sizeMatrix, IPC_CREAT|0600);
145     shmIdM = shmget(IPC_PRIVATE, sizeMatrix, IPC_CREAT|0600);
146
147     matrix1 = shmat(shmId1, NULL, 0);
148     matrix2 = shmat(shmId2, NULL, 0);
149     matrix3 = shmat(shmId3, NULL, 0);
150     matrix4 = shmat(shmId4, NULL, 0);
151     matrix5 = shmat(shmId5, NULL, 0);
152     matrixM = shmat(shmIdM, NULL, 0);
153
154     //crea los espacios de las matrices
155     creaM((void*)matrix1);
156     creaM((void*)matrix2);

```



```

157     creaM((void*)matrix3);
158     creaM((void*)matrix4);
159     creaM((void*)matrix5);
160     creaM((void*)matrixM);
161
162     //llena las matrices
163     llenaM(matrix1);
164     llenaM(matrix2);
165     llenaM(matrix3);
166     llenaM(matrix4);
167
168
169     printf("\n Matrices a multiplicar: \n");
170     showM(matrix1);
171     showM(matrix2);
172
173     if(fork()){
174         for(int i = 0 ; i < 10 ; i++){
175             for(int j = 0 ; j < 10 ; j++){
176                 matrixM[i][j] = (matrix1[i][0]*matrix2[0][j]) + (matrix1[i][1]*matrix2[1][j]) + (matrix1[i][2]*matrix2[2][j])
177                 + (matrix1[i][3]*matrix2[3][j])+(matrix1[i][4]*matrix2[4][j]) + (matrix1[i][5]*matrix2[5][j])
178                 + (matrix1[i][6]*matrix2[6][j]) + (matrix1[i][7]*matrix2[7][j]) + (matrix1[i][8]*matrix2[8][j])
179                 + (matrix1[i][9]*matrix2[9][j]);
180             }
181         }
182     }
183     wait(NULL);
184     //shmdt(matrixM);
185     shmctl(shmIdM, IPC_RMID, 0);
186
187     printf("\n Matrices a sumar: \n");
188     showM(matrix3);
189     showM(matrix4);
190
191     pid1 = fork();
192     if(pid1==0){
193         for(int i = 0 ; i < 10 ; i++){
194             for(int j = 0 ; j < 10 ; j++){
195                 matrix5[i][j] = matrix3[i][j] + matrix4[i][j];
196             }
197         }
198         wait(NULL);
199         //shmdt(matrix5);
200         shmctl(shmIdS, IPC_RMID, 0);
201     }
202     else{
203         sleep(4);
204         printf("\n Matriz Sumada \n");
205         showM(matrix5);
206         //shmdt(matrix5);
207         flag = 1;
208     }
209 }
210
211 else{
212     sleep(4);
213     printf("\n Matriz Multiplicada \n");
214     showM(matrixM);
215     shmdt(matrixM);
216 }
217 }
218
219 if(flag ==1){
220
221     float transf[10][10];
222     conversionMat(matrixM, transf);
223     //showMT(transf);
224
225     FILE* archivo = fopen("invMulti.txt", "w");
226     float d1 = determinant(transf, 10);
227     printf("\nDeterminante de la Matriz de Multiplicación = %f\n", d1);
228
229     if(d1 != 0)
230         cofactor(transf, 10, archivo);
231     else
232         fprintf(archivo, "La matriz no tiene inversa\n");
233
234     fclose(archivo);
235
236     float transfs[10][10];
237     conversionMat(matrixS, transfs);
238     //showMT(transfs);
239
240     FILE* archivo1 = fopen("invSuma.txt", "w");
241     float d2 = determinant(transfs, 10);
242     printf("\nDeterminante de la Matriz de Suma = %f\n", d2);
243
244     if(d1 != 0)
245         cofactor(transfs, 10, archivo);
246     else
247         fprintf(archivo, "La matriz no tiene inversa\n");
248 }
249

```

```

250     fclose(archivo1);
251
252
253 }
254 shmdt(matrixM);
255 shmdt(matrixS);
256 return i;
257
258 }
259
260 void showM(double **matrix){
261     printf("\n");
262     for(int i=0; i<10; i++){
263         for(int j=0; j<10; j++){
264             printf("%.2f\t",matrix[i][j]);
265             printf("\n");
266         }
267     }
268 }
269
270 void creaM(void **m){
271     int i;
272     size_t tamFila = 10 * sizeof(double);
273     m[0] = m+10;
274     for(i=1; i<10; i++){
275         m[i] = (m[i-1]+tamFila);
276     }
277 }
278
279 void llenaM(double **matrix){
280
281     for(int i=0 ; i<10 ; i++){
282         for(int j = 0; j < 10; j++){
283             matrix[i][j] = rand() % (9-1+1)+1;
284         }
285     }
286 }
287
288 void showMT(float mN[][10]){
289
290     for(int i=0 ; i<10 ; i++){
291         for(int j = 0; j < 10; j++){
292             printf("%f ", mN[i][j]);
293         }
294         printf("\n");
295     }
296 }

```

Compilación y ejecución del programa

```

oliver@oliver-VirtualBox:~/Documents/practica5$ ./inciso7.out

```

Matrices a multiplicar:

8.00	1.00	4.00	7.00	5.00	2.00	8.00	5.00	7.00	2.00
1.00	6.00	2.00	7.00	1.00	4.00	8.00	6.00	7.00	5.00
1.00	4.00	7.00	6.00	1.00	2.00	4.00	8.00	3.00	7.00
8.00	1.00	7.00	9.00	7.00	9.00	1.00	3.00	4.00	7.00
2.00	5.00	1.00	3.00	9.00	9.00	5.00	5.00	5.00	9.00
1.00	6.00	3.00	5.00	9.00	1.00	4.00	3.00	8.00	5.00
8.00	4.00	3.00	3.00	2.00	8.00	3.00	2.00	1.00	4.00
7.00	1.00	6.00	7.00	3.00	3.00	4.00	5.00	8.00	9.00
4.00	6.00	3.00	4.00	8.00	2.00	5.00	3.00	2.00	1.00
5.00	9.00	3.00	7.00	1.00	4.00	5.00	1.00	5.00	6.00
4.00	2.00	6.00	8.00	7.00	6.00	1.00	1.00	9.00	6.00
7.00	3.00	2.00	7.00	7.00	1.00	8.00	9.00	1.00	8.00
7.00	5.00	5.00	9.00	2.00	5.00	3.00	7.00	5.00	8.00
1.00	7.00	7.00	4.00	3.00	2.00	7.00	3.00	3.00	6.00
9.00	9.00	7.00	8.00	7.00	2.00	8.00	3.00	1.00	6.00
1.00	7.00	8.00	4.00	7.00	1.00	8.00	7.00	5.00	2.00
5.00	3.00	6.00	1.00	6.00	8.00	2.00	3.00	1.00	2.00
7.00	7.00	9.00	4.00	6.00	4.00	5.00	2.00	6.00	3.00
6.00	5.00	7.00	4.00	8.00	4.00	2.00	4.00	9.00	6.00
5.00	2.00	6.00	1.00	2.00	2.00	6.00	2.00	3.00	5.00

Matriz Multiplicada

248.00	245.00	324.00	241.00	279.00	211.00	200.00	161.00	236.00	247.00
229.00	227.00	297.00	185.00	259.00	168.00	234.00	195.00	189.00	227.00
227.00	211.00	273.00	194.00	198.00	151.00	216.00	175.00	177.00	225.00
254.00	301.00	372.00	298.00	286.00	175.00	295.00	216.00	263.00	296.00
278.00	282.00	347.00	234.00	304.00	153.00	309.00	216.00	198.00	250.00
268.00	241.00	276.00	226.00	256.00	142.00	242.00	185.00	168.00	254.00
165.00	174.00	237.00	198.00	215.00	129.00	192.00	161.00	178.00	188.00
262.00	249.00	347.00	246.00	267.00	198.00	233.00	181.00	266.00	280.00
220.00	203.00	228.00	215.00	224.00	133.00	204.00	160.00	131.00	208.00
216.00	197.00	261.00	217.00	248.00	150.00	236.00	208.00	185.00	255.00

Matrices a sumar:

3.00	9.00	2.00	9.00	3.00	5.00	3.00	5.00	5.00	7.00
7.00	1.00	9.00	4.00	2.00	7.00	6.00	4.00	8.00	5.00
7.00	2.00	6.00	4.00	9.00	6.00	3.00	6.00	5.00	5.00
1.00	7.00	2.00	9.00	7.00	2.00	5.00	7.00	6.00	7.00
4.00	1.00	5.00	3.00	3.00	6.00	7.00	8.00	7.00	6.00
1.00	3.00	7.00	4.00	6.00	6.00	7.00	6.00	2.00	2.00
9.00	9.00	7.00	1.00	9.00	4.00	1.00	2.00	1.00	4.00
8.00	3.00	5.00	3.00	5.00	5.00	8.00	3.00	3.00	4.00
6.00	1.00	6.00	3.00	2.00	9.00	6.00	9.00	5.00	6.00
1.00	4.00	5.00	5.00	3.00	2.00	8.00	3.00	3.00	7.00

6.00	1.00	7.00	8.00	3.00	2.00	3.00	9.00	2.00	3.00
3.00	7.00	3.00	6.00	9.00	3.00	5.00	4.00	2.00	9.00
9.00	9.00	4.00	2.00	5.00	4.00	4.00	1.00	6.00	6.00
5.00	9.00	7.00	2.00	6.00	7.00	4.00	8.00	4.00	5.00
9.00	4.00	1.00	9.00	9.00	9.00	2.00	4.00	3.00	1.00
4.00	9.00	1.00	5.00	2.00	3.00	8.00	5.00	3.00	2.00
1.00	6.00	8.00	5.00	7.00	4.00	1.00	8.00	1.00	4.00
2.00	9.00	6.00	2.00	8.00	5.00	8.00	1.00	7.00	2.00
8.00	8.00	8.00	8.00	3.00	9.00	1.00	8.00	4.00	2.00
7.00	3.00	7.00	5.00	5.00	4.00	7.00	5.00	1.00	7.00

Matriz Sumada

9.00	10.00	9.00	17.00	6.00	7.00	6.00	14.00	7.00	10.00
10.00	8.00	12.00	10.00	11.00	10.00	11.00	8.00	10.00	14.00
16.00	11.00	10.00	6.00	14.00	10.00	7.00	7.00	11.00	11.00
6.00	16.00	9.00	11.00	13.00	9.00	9.00	15.00	10.00	12.00
13.00	5.00	6.00	12.00	12.00	15.00	9.00	12.00	10.00	7.00
5.00	12.00	8.00	9.00	8.00	9.00	15.00	11.00	5.00	4.00
10.00	15.00	15.00	6.00	16.00	8.00	2.00	10.00	2.00	8.00
10.00	12.00	11.00	5.00	13.00	10.00	16.00	4.00	10.00	6.00
14.00	9.00	14.00	11.00	5.00	18.00	7.00	17.00	9.00	8.00
8.00	7.00	12.00	10.00	8.00	6.00	15.00	8.00	4.00	14.00

Determinante de la Matriz de Multiplicación = -1383374606303232.000000

Determinante de la matriz de Suma = 908883456.000000

Archivos creados:



invMulti.txt



invSuma.txt

Contenido de los archivos, los cuales contienen la matriz inversa

InvMulti.txt

```
1 The inverse of matrix is :
2 0.649656 -0.109013 0.017987 0.194511 0.348781 -0.742316 -0.024656 -0.055592 0.073575 -0.140685
3 0.117293 0.013687 0.014526 0.013755 0.076321 -0.152137 -0.013792 -0.025674 -0.005213 -0.009430
4 0.190355 -0.031382 -0.002001 0.064225 0.123539 -0.226963 -0.006711 -0.026023 0.040673 -0.058486
5 -0.040689 0.008914 -0.002065 -0.008551 0.011160 0.042974 -0.000551 -0.002671 -0.001386 -0.019936
6 -0.063343 -0.000636 -0.005835 -0.012233 -0.036595 0.080804 0.028980 0.008173 -0.005867 -0.011587
7 0.484845 -0.055200 0.010543 0.158244 0.279573 -0.550330 -0.017133 -0.054659 0.073763 -0.170593
8 -0.113938 0.017881 0.015895 -0.019749 -0.069178 0.127495 -0.007602 0.009463 -0.032403 0.031853
9 -0.309419 0.041922 -0.042504 -0.094359 -0.178507 0.369024 0.010742 0.045576 -0.006745 0.073610
10 -0.154282 0.005053 -0.033496 -0.041281 -0.121034 0.220943 0.031455 0.026608 0.002907 0.021653
11 -0.686265 0.094452 0.029785 -0.221482 -0.388553 0.755121 0.004852 0.065615 -0.125704 0.236718
```

invSuma.txt

```
1 The inverse of matrix is :
2 0.087129 -0.931637 0.058062 0.578982 -0.220685 0.846665 0.533384 0.168390 -0.344510 -0.424603
3 -0.027679 0.013073 -0.025940 0.034942 0.024004 -0.092668 0.011443 0.001857 0.006514 0.057824
4 -0.022807 0.654144 -0.158042 -0.413048 0.105120 -0.524411 -0.295605 0.000642 0.217418 0.249190
5 -0.077360 -0.124411 0.133708 -0.034149 0.014957 0.289402 0.113403 -0.200972 -0.174469 0.078074
6 -0.004206 0.118943 -0.022151 -0.073250 0.005929 -0.044202 0.012367 0.115254 0.020256 -0.124983
7 0.097140 -0.373201 -0.007363 0.328163 -0.097714 0.323048 0.152978 0.213366 -0.200295 -0.278863
8 -0.081112 0.300513 0.042518 -0.293651 0.052933 -0.244591 -0.172054 -0.101974 0.123771 0.271113
9 -0.027727 0.123896 0.144649 -0.216289 0.043009 -0.087738 -0.039722 -0.197099 0.034915 0.140348
10 -0.005827 0.529141 -0.098236 -0.348843 0.191120 -0.601387 -0.434108 -0.051817 0.311674 0.308026
11 0.065415 -0.160364 -0.064229 0.343684 -0.069358 0.002840 0.045532 0.027853 0.065982 -0.181088
```

2.7.2 Codificación en Windows

Código fuente

```
C Matrices.c • C Servidor.c C Cliente.c C ProcesoH.c C ProcesoN.c
C: > Users > rodri > Desktop > Escuela > Sistemas > Practica5 > C Matrices.c > ...
1  #include <windows.h>
2  #include "Matrices.h"
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <unistd.h>
6  #include <math.h>
7
8  void SumaM(int **m1, int **m2, int size_m, int ***MatrizR)
9  {
10     (*MatrizR) = (int **)calloc(size_m, sizeof(int *));
11     for (int i = 0; i < size_m; i++)
12     {
13         (*MatrizR)[i] = (int *)calloc(size_m, sizeof(int));
14     }
15     for (int i = 0; i < size_m; i++)
16     {
17         for (int j = 0; j < size_m; j++)
18         {
19             (*MatrizR)[i][j] = m1[i][j] + m2[i][j];
20         }
21     }
22 }
23
24 void MultiplicacionM(int **m1, int **m2, int size_m, int ***MatrizR)
25 {
26     (*MatrizR) = (int **)calloc(size_m, sizeof(int *));
27     for (int i = 0; i < size_m; i++)
28     {
29         (*MatrizR)[i] = (int *)calloc(size_m, sizeof(int));
30     }
31     for (int i = 0; i < size_m; i++)
32     {
33         for (int j = 0; j < size_m; j++)
34         {
35             int suma = 0;
36             for (int k = 0; k < size_m; k++)
37             {
38                 suma += m1[j][k] * m2[k][i];
39             }
40         }
41     }
42 }
```

```
38         suma += m1[j][k] * m2[k][i];
39     }
40
41     (*MatrizR)[j][i] = suma;
42 }
43 }
44 }
45
46 void ImprimirMatriz(int **m1, int size_m)
47 {
48     printf("La matriz es : \n");
49     for (int i = 0; i < size_m; i++)
50     {
51         for (int j = 0; j < size_m; j++)
52         {
53             printf("%d\t", m1[i][j]);
54         }
55         printf("\n");
56     }
57 }
58
59 void AsignarMatriz(int ***m1, int size_m)
60 {
61     (*m1) = (int **)calloc(size_m, sizeof(int *));
62     for (int i = 0; i < size_m; i++)
63     {
64         (*m1)[i] = (int *)calloc(size_m, sizeof(int));
65     }
66
67     for (int i = 0; i < size_m; i++)
68     {
69         for (int j = 0; j < size_m; j++)
70         {
71             (*m1)[i][j] = j;
72         }
73     }
74 }
75
76 void AsignarMatrizN(int ***m1, int size_m, int *mN, int start)
77 {
78     for (int i = 0; i < size_m; i++)
79     {
80         for (int j = 0; j < size_m; j++)
81         {
82             (*m1)[i][j] = mN[j];
83         }
84     }
85 }
```

```

75 void AsignarMatrizN(int ***m1, int size_m, int *mN, int start)
76 {
77     int k=start;
78     (*m1) = (int **)calloc(size_m, sizeof(int *));
79     for (int i = 0; i < size_m; i++)
80     {
81         (*m1)[i] = (int *)calloc(size_m, sizeof(int));
82     }
83
84     for (int i = 0; i < size_m; i++)
85     {
86         for (int j = 0; j < size_m; j++)
87         {
88             (*m1)[i][j] = mN[k];
89             k++;
90         }
91     }
92 }
93
94 float determinant(float a[10][10], float k) {
95     float s = 1, det = 0, b[10][10];
96     int i, j, m, n, c;
97     if (k == 1)
98     {
99         return (a[0][0]);
100     }
101     else
102     {
103         det = 0;
104         for (c = 0; c < k; c++)
105         {
106             m = 0;
107             n = 0;
108             for (i = 0; i < k; i++)
109             {
110                 for (j = 0; j < k; j++)
111                 {
112                     b[i][j] = 0;
113
114                     if (i != 0 && j != c)
115                     {
116                         b[m][n] = a[i][j];
117                         if (n < (k - 2))
118                             n++;
119                         else
120                         {
121                             n = 0;
122                             m++;
123                         }
124                     }
125                 }
126                 det = det + s * (a[0][c] * determinant(b, k - 1));
127                 s = -1 * s;
128             }
129         }
130         return (det);
131     }
132 }
133
134 void transpose(float num[10][10], float fac[10][10], float r, FILE* archivo){
135     int i, j;
136     r = 10;
137
138     float b[10][10], inverse[10][10], d;
139
140     for (i = 0; i < r; i++)
141     {
142         for (j = 0; j < r; j++)
143         {
144             b[i][j] = fac[j][i];
145         }
146     }
147     d = determinant(num, r);
148     for (i = 0; i < r; i++)
149     {
150         for (j = 0; j < r; j++)
151         {
152             inverse[i][j] = b[i][j] / d;
153
154             inverse[i][j] = 0[i][j] / d;
155         }
156     }
157
158     fprintf(archivo, "The inverse of matrix is : \n");
159
160     for (i = 0; i < r; i++) {
161         for (j = 0; j < r; j++) {
162             fprintf(archivo, "%f\t", inverse[i][j]);
163         }
164         fprintf(archivo, "\n");
165     }
166 }
167
168 void cofactor(float num[10][10], float f, FILE* archivo){
169     float b[10][10], fac[10][10];
170     int p, q, m, n, i, j;
171     for (q = 0; q < f; q++)
172     {
173         for (p = 0; p < f; p++)
174         {
175             m = 0;
176             n = 0;
177             for (i = 0; i < f; i++)
178             {
179                 for (j = 0; j < f; j++)
180                 {
181                     if (i != q && j != p)
182                     {
183                         b[m][n] = num[i][j];
184                         if (n < (f - 2))
185                             n++;
186                         else
187                         {
188                             n = 0;
189                             m++;
190                         }
191                     }
192                 }
193                 fac[q][p] = pow(-1, q + p) * determinant(b, f - 1);
194             }
195         }
196         transpose(num, fac, f, archivo);
197     }
198 }
199
200 void conversionMat(int** origi, float transf[][10]){
201
202     for(int i=0; i<10; i++){
203         for(int j=0; j<10; j++){
204             transf[i][j]= (float) origi[i][j];
205         }
206     }
207 }
208
209 void ImprimirArchivoIM(char* path, float inversa1[10][10]){
210     FILE *archivo = fopen(path, "w");
211     for (int i = 0; i < 10; i++)
212     {
213         for (int j = 0; j < 10; j++)
214         {
215             fprintf(archivo, "%.2f\t", inversa1[i][j]);
216         }
217         fprintf(archivo, "\n");
218     }
219
220     fclose(archivo);
221 }

```



```

C Matrices.c ● C Servidor.c X C Cliente.c C ProcesoH.c C ProcesoN.c C MCLie
C: > Users > rodri > Desktop > Escuela > Sistemas > Practica5 > C Servidor.c > main(void)
1 #include <windows.h> /* Servidor de la memoria compartida */
2 #include <stdio.h> /* (ejecutar el servidor antes de ejecutar el cliente)*/
3 #include "Matrices.h"
4 #define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
5 int main(void)
6 {
7     HANDLE hArchMapeo;
8     char *idMemCompartida = "MemoriaCompartida";
9     int *apDatos, *apTrabajo, c;
10    if ((hArchMapeo = CreateFileMapping(
11        INVALID_HANDLE_VALUE, // usa memoria compartida
12        NULL, // seguridad por default
13        PAGE_READWRITE, // acceso lectura/escritura a la memoria
14        0, // tamaño maximo parte alta de un DWORD
15        TAM_MEM, // tamaño maximo parte baja de un DWORD
16        idMemCompartida) // identificador de la memoria compartida
17    ) == NULL)
18    {
19        printf("No se mapeo la memoria compartida: (%i)\n", GetLastError());
20        exit(-1);
21    }
22    if ((apDatos = (int *)MapViewOfFile(hArchMapeo, // Manejador del mapeo
23        FILE_MAP_ALL_ACCESS, // Permiso de lectura/escritura en la memoria
24        0,
25        0,
26        TAM_MEM)) == NULL)
27    {
28        printf("No se creo la memoria compartida: (%i)\n", GetLastError());
29        CloseHandle(hArchMapeo);
30        exit(-1);
31    }
32    int **matriz1;
33    int **matriz2;
34    int **matriz3;
35    int **matriz4;
36    apTrabajo = apDatos;
37    int fin_intervalo = 10;
38
39    for (c = 0; c < 200; c++){
40        *apTrabajo++ = rand() % fin_intervalo;
41    }
42
43    *apTrabajo = 999;
44
45    while (*apDatos != 991)
46        Sleep(1);
47    *apDatos = 0;
48    //////////////////////////////////////
49    STARTUPINFO si; /* Estructura de información inicial para Windows */
50    PROCESS_INFORMATION pi; /* Estructura de información del adm. de procesos */
51    ZeroMemory(&si, sizeof(si));
52    si.cb = sizeof(si);
53    ZeroMemory(&pi, sizeof(pi));
54    // Creación proceso hijo
55    if (!CreateProcess(NULL,
56        "C:\\Users\\rodri\\Desktop\\Escuela\\Sistemas\\Practica5\\ProcesoH.exe",
57        NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
58    {
59        printf("Fallo al invocar CreateProcess (%d)\n", GetLastError());
60        return 0;
61    }
62    // Proceso padre
63
64    WaitForSingleObject(pi.hProcess, INFINITE);
65    // Terminación controlada del proceso e hilo asociado de ejecución
66    CloseHandle(pi.hProcess);
67    CloseHandle(pi.hThread);
68    //////////////////////////////////////
69    AsignarMatrizN(&matriz1, 10, apDatos, 0);
70    AsignarMatrizN(&matriz2, 10, apDatos, 100);
71    AsignarMatrizN(&matriz3, 10, apDatos, 200);
72    AsignarMatrizN(&matriz4, 10, apDatos, 300);
73    ImprimirMatriz(matriz1, 10);
74    ImprimirMatriz(matriz2, 10);
75    ImprimirMatriz(matriz3, 10);
76    ImprimirMatriz(matriz4, 10);
77    //Convertimos matriz de int a float
78    float MatrizOut1[10][10];
79    float MatrizOut2[10][10];

```

```

77 //Convertimos matriz de int a float
78 float MatrizOut1[10][10];
79 float MatrizOut2[10][10];
80 float inversa1[10][10];
81 float inversa2[10][10];
82 conversionMat(matriz1, MatrizOut1);
83 conversionMat(matriz2, MatrizOut2);
84 conversionMat(matriz3, inversa1);
85 conversionMat(matriz4, inversa2);
86 printf("Soy el abuelo\n");
87 ImprimirArchivoIM("PrimerMatriz.txt", MatrizOut1);
88 ImprimirArchivoIM("SegundaMatriz.txt", MatrizOut2);
89 ImprimirArchivoM("Inversa1Matriz.txt", inversa1);
90 ImprimirArchivoM("Inversa2Matriz.txt", inversa2);
91
92
93 UnmapViewOfFile(apDatos);
94 CloseHandle(hArchMapeo);
95 exit(0);
96

```

```

Matrices.c • C Servidor.c • C Cliente.c • C ProcesoH.c • C ProcesoN.c • MCliente.c
> Users > rodri > Desktop > Escuela > Sistemas > Practica5 > C Cliente.c > main(void)
1  #include <windows.h> /* Cliente de la memoria compartida */
2  #include <stdio.h>
3  #define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
4  int main(void)
5  {
6      HANDLE hArchMapeo;
7      char *idMemCompartida = "MemoriaCompatida";
8      int *apDatos, *apTrabajo, c;
9      if ((hArchMapeo = OpenFileMapping(
10         FILE_MAP_ALL_ACCESS, // acceso lectura/escritura de la memoria compartida
11         FALSE,                // no se hereda el nombre
12         idMemCompartida)      // identificador de la memoria compartida
13         ) == NULL)
14      {
15          printf("No se abrio archivo de mapeo de la memoria compartida: (%i)\n", GetLastError());
16          exit(-1);
17      }
18      if ((apDatos = (int *)MapViewOfFile(hArchMapeo, // Manejador del mapeo
19          FILE_MAP_ALL_ACCESS, // Permiso de lectura/escritura en la memoria
20          0,
21          0,
22          TAM_MEM)) == NULL)
23      {
24          printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
25          CloseHandle(hArchMapeo);
26          exit(-1);
27      }
28      apTrabajo = apDato
29      *apDatos = 991;
30      UnmapViewOfFile(apDatos);
31      CloseHandle(hArchMapeo);
32      exit(0);
33  }

```



```

C Matrices.c • C Servidor.c • C Cliente.c • C ProcesoH.c • C ProcesoN.c • MCliente.c
C: > Users > rodri > Desktop > Escuela > Sistemas > Practica5 > C ProcesoH.c > main(void)
1  #include <windows.h> /* Cliente de la memoria compartida */
2  #include <stdio.h>
3  #include "Matrices.h"
4  #define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
5  int main(void)
6  {
7      HANDLE hArchMapeo;
8      char *idMemCompartida = "MemoriaCompatida";
9      int *apDatos, *apTrabajo, c;
10     if ((hArchMapeo = OpenFileMapping(
11         FILE_MAP_ALL_ACCESS, // acceso lectura/escritura de la memoria compartida
12         FALSE, // no se hereda el nombre
13         idMemCompartida) // identificador de la memoria compartida
14     ) == NULL)
15     {
16         printf("No se abrio archivo de mapeo de la memoria compartida: (%i)\n", GetLastError());
17         exit(-1);
18     }
19     if ((apDatos = (int *)MapViewOfFile(hArchMapeo, // Manejador del mapeo
20         FILE_MAP_ALL_ACCESS, // Permiso de lectura/escritura en la memoria
21         0,
22         0,
23         TAM_MEM)) == NULL)
24     {
25         printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
26         CloseHandle(hArchMapeo);
27         exit(-1);
28     }
29
30     int **matriz1;
31     int **matriz2;
32     int **Result;
33     apTrabajo = apDatos;
34     AsignarMatrizN(&matriz1, 10, apDatos, 0);
35     AsignarMatrizN(&matriz2, 10, apDatos, 100);
36     MultiplicacionM(matriz1, matriz2, 10, &Result);
37     int k=200;
38     for (int i = 0; i < 10; i++)
39     {
40         for (int j = 0; j < 10; j++)
41         {
42             apTrabajo[k++] = Result[i][j];
43         }
44     }
45     //////////////////////////////////////
46     STARTUPINFO si; /* Estructura de informaci3n inicial para Windows */
47     PROCESS_INFORMATION pi; /* Estructura de informaci3n del adm. de procesos */
48     ZeroMemory(&si, sizeof(si));
49     si.cb = sizeof(si);
50     ZeroMemory(&pi, sizeof(pi));
51     // Creaci3n proceso hijo
52     if (!CreateProcess(NULL,
53         "C:\\Users\\rodri\\Desktop\\Escuela\\Sistemas\\Practica5\\ProcesoN.exe",
54         NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
55     {
56         printf("Fallo al invocar CreateProcess (%d)\n", GetLastError());
57         return 0;
58     }
59     // Proceso padre
60     printf("Soy el padre\n");
61     WaitForSingleObject(pi.hProcess, INFINITE);
62     // Terminaci3n controlada del proceso e hilo asociado de ejecuci3n
63     CloseHandle(pi.hProcess);
64     CloseHandle(pi.hThread);
65     //////////////////////////////////////
66     UnmapViewOfFile(apDatos);
67     CloseHandle(hArchMapeo);
68     exit(0);
69 }

```

```

Matrices.c • C Servidor.c • C Cliente.c • C ProcesoH.c • C ProcesoN.c • MCliente.c
> Users > rodrigo > Desktop > Escuela > Sistemas > Practica5 > C ProcesoN.c > main(void)
1  #include <windows.h> /* Cliente de la memoria compartida */
2  #include <stdio.h>
3  #include "Matrices.h"
4  #define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
5  int main(void)
6  {
7      HANDLE hArchMapeo;
8      char *idMemCompartida = "MemoriaCompartida";
9      int *apDatos, *apTrabajo, c;
10     if ((hArchMapeo = OpenFileMapping(
11         FILE_MAP_ALL_ACCESS, // acceso lectura/escritura de la memoria compartida
12         FALSE, // no se hereda el nombre
13         idMemCompartida) // identificador de la memoria compartida
14     ) == NULL)
15     {
16         printf("No se abrio archivo de mapeo de la memoria compartida: (%i)\n", GetLastError());
17         exit(-1);
18     }
19     if ((apDatos = (int *)MapViewOfFile(hArchMapeo, // Manejador del mapeo
20         FILE_MAP_ALL_ACCESS, // Permiso de lectura/escritura en la memoria
21         0,
22         0,
23         TAM_MEM)) == NULL)
24     {
25         printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
26         CloseHandle(hArchMapeo);
27         exit(-1);
28     }
29
30     int **matriz1;
31     int **matriz2;
32     int **Result;
33     apTrabajo = apDatos;
34     AsignarMatrizN(&matriz1, 10, apDatos, 0);
35     AsignarMatrizN(&matriz2, 10, apDatos, 100);
36     SumaM(matriz1, matriz2, 10, &Result);
37     int k = 300;
38     for (int i = 0; i < 10; i++)
39     {
40         for (int j = 0; j < 10; j++)
41         {
42             apTrabajo[k++] = Result[i][j];
43         }
44     }
45     *apTrabajo = 999;
46     UnmapViewOfFile(apDatos);
47     CloseHandle(hArchMapeo);
48     exit(0);

```

Compilación y ejecución del programa

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\rodri\Desktop\Escuela\Sistemas\Practica5> ./Cliente
PS C:\Users\rodri\Desktop\Escuela\Sistemas\Practica5> █
```

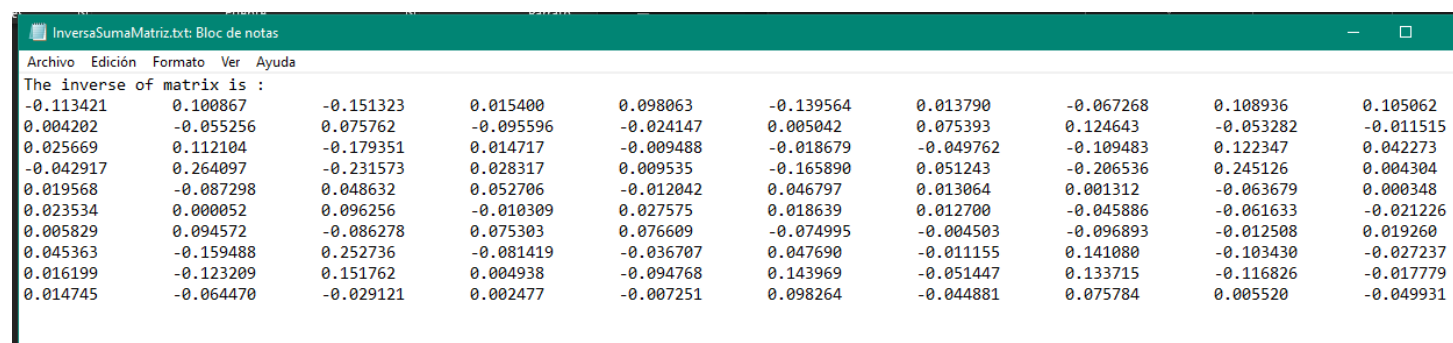
```
Matriz 1
La matriz es :
0 7 4 0 9 4 8 8 2 4
5 5 1 7 1 1 5 2 7 6
1 4 2 3 2 2 1 6 8 5
7 6 1 8 9 2 7 9 5 4
3 1 2 3 3 4 1 1 3 8
7 4 2 7 7 9 3 1 9 8
6 5 0 2 8 6 0 2 4 8
6 5 0 9 0 0 6 1 3 8
9 3 4 4 6 0 6 6 1 8
4 9 6 3 7 8 8 2 9 1

Matriz 2
La matriz es :
3 5 9 8 4 0 7 6 3 6
1 5 4 2 0 9 7 3 7 2
6 0 1 6 5 7 5 4 1 2
0 0 1 4 6 0 7 1 7 7
7 7 3 3 5 9 9 8 1 8
2 6 6 0 3 8 0 1 2 5
0 9 4 7 8 3 5 1 2 0
1 6 4 0 6 1 8 9 8 4
1 4 3 9 8 8 0 8 7 7
8 3 8 3 7 1 0 7 3 4

Matriz Multiplicacion
La matriz es :
144 262 185 151 233 256 254 237 176 176
92 166 179 203 225 148 174 186 196 185
91 143 140 140 191 162 137 201 181 165
146 289 246 239 310 232 340 302 266 281
119 116 150 117 161 118 99 150 105 146
178 251 270 251 300 275 222 274 226 303
161 199 220 150 194 207 179 229 155 222
91 151 184 187 212 96 178 155 181 166
167 220 234 211 258 149 264 255 177 207
141 285 225 259 278 349 261 247 223 246

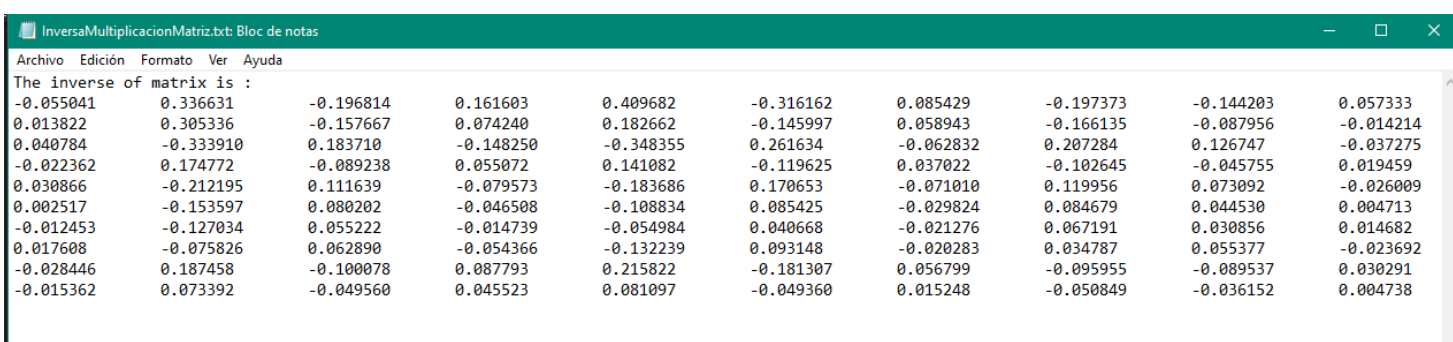
Matriz Suma
La matriz es :
3 12 13 8 13 4 15 14 5 10
6 10 5 9 1 10 12 5 14 8
7 4 3 9 7 9 6 10 9 7
7 6 2 12 15 2 14 10 12 11
10 8 5 6 8 13 10 9 4 16
9 10 8 7 10 17 3 2 11 13
6 14 4 9 16 9 5 3 6 8
7 11 4 9 6 1 14 10 11 12
10 7 7 13 14 8 6 14 8 15
12 12 14 6 14 9 8 9 12 5
```

Archivos generados donde se encuentran las inversas de la matriz multiplicación y de la matriz suma



The inverse of matrix is :

-0.113421	0.100867	-0.151323	0.015400	0.098063	-0.139564	0.013790	-0.067268	0.108936	0.105062
0.004202	-0.055256	0.075762	-0.095596	-0.024147	0.005042	0.075393	0.124643	-0.053282	-0.011515
0.025669	0.112104	-0.179351	0.014717	-0.009488	-0.018679	-0.049762	-0.109483	0.122347	0.042273
-0.042917	0.264097	-0.231573	0.028317	0.009535	-0.165890	0.051243	-0.206536	0.245126	0.004304
0.019568	-0.087298	0.048632	0.052706	-0.012042	0.046797	0.013064	0.001312	-0.063679	0.000348
0.023534	0.000052	0.096256	-0.010309	0.027575	0.018639	0.012700	-0.045886	-0.061633	-0.021226
0.005829	0.094572	-0.086278	0.075303	0.076609	-0.074995	-0.004503	-0.096893	-0.012508	0.019260
0.045363	-0.159488	0.252736	-0.081419	-0.036707	0.047690	-0.011155	0.141080	-0.103430	-0.027237
0.016199	-0.123209	0.151762	0.004938	-0.094768	0.143969	-0.051447	0.133715	-0.116826	-0.017779
0.014745	-0.064470	-0.029121	0.002477	-0.007251	0.098264	-0.044881	0.075784	0.005520	-0.049931



The inverse of matrix is :

-0.055041	0.336631	-0.196814	0.161603	0.409682	-0.316162	0.085429	-0.197373	-0.144203	0.057333
0.013822	0.305336	-0.157667	0.074240	0.182662	-0.145997	0.058943	-0.166135	-0.087956	-0.014214
0.040784	-0.333910	0.183710	-0.148250	-0.348355	0.261634	-0.062832	0.207284	0.126747	-0.037275
-0.022362	0.174772	-0.089238	0.055072	0.141082	-0.119625	0.037022	-0.102645	-0.045755	0.019459
0.030866	-0.212195	0.111639	-0.079573	-0.183686	0.170653	-0.071010	0.119956	0.073092	-0.026009
0.002517	-0.153597	0.080202	-0.046508	-0.108834	0.085425	-0.029824	0.084679	0.044530	0.004713
-0.012453	-0.127034	0.055222	-0.014739	-0.054984	0.040668	-0.021276	0.067191	0.030856	0.014682
0.017608	-0.075826	0.062890	-0.054366	-0.132239	0.093148	-0.020283	0.034787	0.055377	-0.023692
-0.028446	0.187458	-0.100078	0.087793	0.215822	-0.181307	0.056799	-0.095955	-0.089537	0.030291
-0.015362	0.073392	-0.049560	0.045523	0.081097	-0.049360	0.015248	-0.050849	-0.036152	0.004738

3 Conclusiones

Como se pudo observar, la comunicación entre procesos es una función indispensable para un Sistema Operativo, se sabe que existen dos métodos, memoria compartida y paso de mensajes, la primera es una de las más fáciles de implementar ya que sigue un patrón de diseño fácil de comprender, conocido como cliente-servidor y con el cual el acceso a la información se realiza de una manera rápida y eficaz, además este tipo de comunicación es escalable ya que puedes hacer n cantidad de nodos que se conecten al servidor, claro está mientras no se acaba la capacidad de la memoria. En comparación con la memoria compartida, el paso de mensajes no es tan escalable, ya que requiere mucho procesamiento para pasar la información lo que lo hace más complicado y tedioso, además en el caso de que se emplee de una manera incorrecta, este puede llegar a causar problemas de ineficiencia, puesto que crear canales de comunicación debe tener un análisis previo para buscar la eficiencia del paso de información.

Cada método tiene su especialidad y función única, la cual se debe analizar bien, en qué casos conviene utilizar una u otra, esto es parte del trabajo de ingeniería, con lo que un programador debe conocer las características diferenciadoras, aunque en la actualidad se hace mayor mención del método de memoria compartida.