

INTRODUCCIÓN

Comunicación Entre Procesos

La comunicación entre procesos, en inglés IPC (Inter-process Communication) es una función básica de los sistemas operativos. Los procesos pueden comunicarse entre sí a través de compartir espacios de memoria, ya sean variables compartidas o buffers, o a través de las herramientas provistas por las rutinas de IPC. El 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.

La comunicación se establece siguiendo una serie de reglas (protocolos de comunicación). Los protocolos desarrollados para internet son los mayormente usados: IP (capa de red), protocolo de control de transmisión (capa de transporte) y protocolo de transferencia de archivos, protocolo de transferencia de hipertexto (capa de aplicación).

Los procesos pueden ejecutarse en una o más computadoras conectadas a una red. Las técnicas de IPC están divididas dentro de métodos para: paso de mensajes, sincronización, memoria compartida y llamadas de procedimientos remotos (RPC). El método de IPC usado puede variar dependiendo del ancho de banda y latencia (el tiempo desde el pedido de información y el comienzo del envío de la misma) de la comunicación entre procesos, y del tipo de datos que están siendo comunicados.

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 la 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 establecerse).
- Empiece a intercambiar mensajes utilizando primitivas básicas.

Necesitamos al menos dos primitivas:

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

Competencia.

Desarrollo.

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:

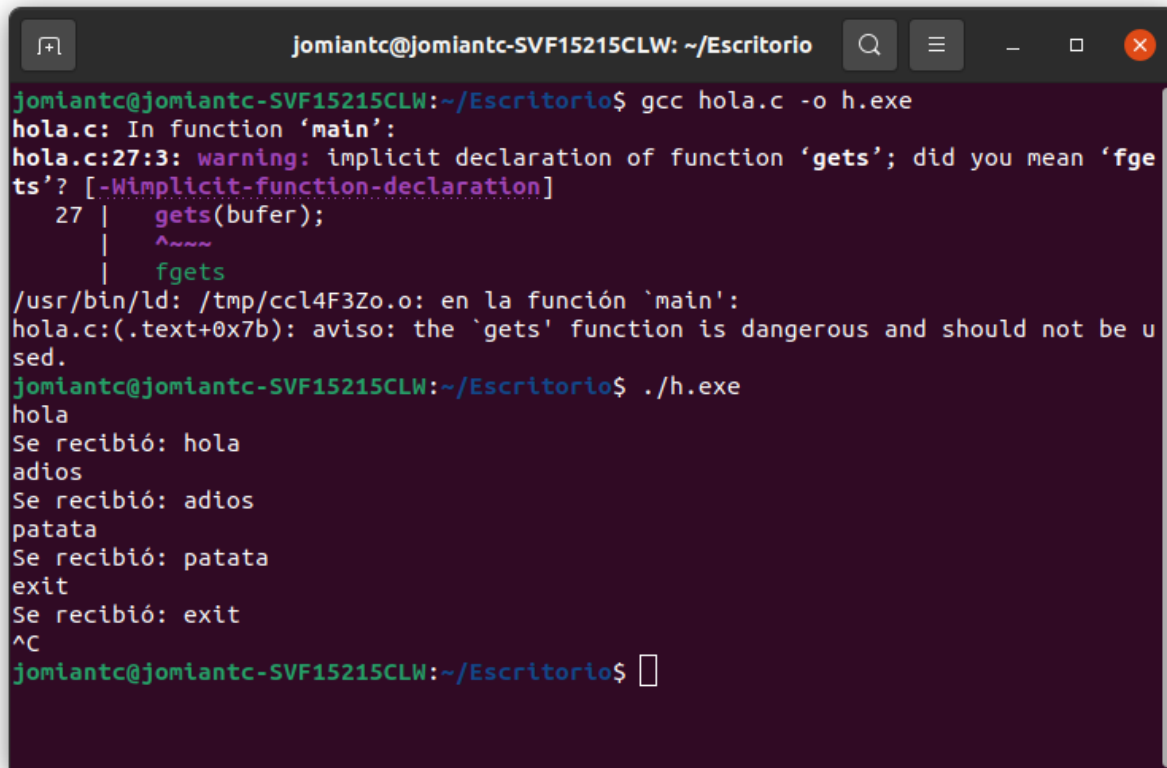
Se retorna un puntero a la zona de memoria compartida

Se retorna -1 en caso de algún error

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

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#define VALOR 1
int main(void)
{
    int desc_arch[2];
    char bufer[100];
    if(pipe(desc_arch) != 0)
        exit(1);
    if(fork() == 0)
    {
        while(VALOR)
        {
            read(desc_arch[0], bufer, sizeof(bufer));
            printf("Se recibió: %s\n", bufer);
        }
    }
    while(VALOR)
    {
        gets(bufer);
        write(desc_arch[1], bufer, strlen(bufer)+1);
    }
}
```

Ejecución



```
jomiantc@jomiantc-SVF15215CLW: ~/Escritorio
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$ gcc hola.c -o h.exe
hola.c: In function 'main':
hola.c:27:3: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-function-declaration]
   27 |     gets(buf);
      |     ^~~~~
      |     fgets
/usr/bin/ld: /tmp/cc14F3Zo.o: en la función 'main':
hola.c:(.text+0x7b): aviso: the 'gets' function is dangerous and should not be used.
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$ ./h.exe
hola
Se recibió: hola
adios
Se recibió: adios
patata
Se recibió: patata
exit
Se recibió: exit
^C
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$
```

El programa funciona perfectamente pero al principio no entendía cómo funcionaba una vez que revise el código lo entendí más claramente, el programa obtiene del buffer en este caso de la consola de comandos, en este caso escribimos una oración y el programa obtiene la cadena mediante el buffer de la consola

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

```
#include "windows.h"
#include "stdio.h"
#include "string.h"
int main (int argc, char *argv[])
{
    char mensaje[]="Tuberias en Windows";
    DWORD escritos;
    HANDLE hLecturaPipe, hEscrituraPipe;
```

```

PROCESS_INFORMATION piHijo;
STARTUPINFO siHijo;
SECURITY_ATTRIBUTES pipeSeg =
{sizeof(SEcurity_ATTRIBUTES), NULL, TRUE};
/* Obtención de información para la inicialización del proceso hijo
*/
GetStartupInfo (&siHijo);
/* Creación de la tubería sin nombre */
CreatePipe (&hLecturaPipe, &hEscrituraPipe, &pipeSeg, 0);
/* Escritura en la tubería sin nombre */
WriteFile(hEscrituraPipe, mensaje, strlen(mensaje)+1, &escritos,
NULL);
siHijo.hStdInput = hLecturaPipe;
siHijo.hStdError = GetStdHandle (STD_ERROR_HANDLE);
siHijo.hStdOutput = GetStdHandle (STD_OUTPUT_HANDLE);
siHijo.dwFlags = STARTF_USESTDHANDLES;
CreateProcess (NULL, argv[1], NULL, NULL,
TRUE, /* Hereda el proceso hijo los manejadores de la tubería del
padre */
0, NULL, NULL, &siHijo, &piHijo);
WaitForSingleObject (piHijo.hProcess, INFINITE);
printf("Mensaje recibido en el proceso hijo, termina el proceso
padre\n");
CloseHandle(hLecturaPipe);
CloseHandle(hEscrituraPipe);
CloseHandle(piHijo.hThread);
CloseHandle(piHijo.hProcess);
return 0;
}

```

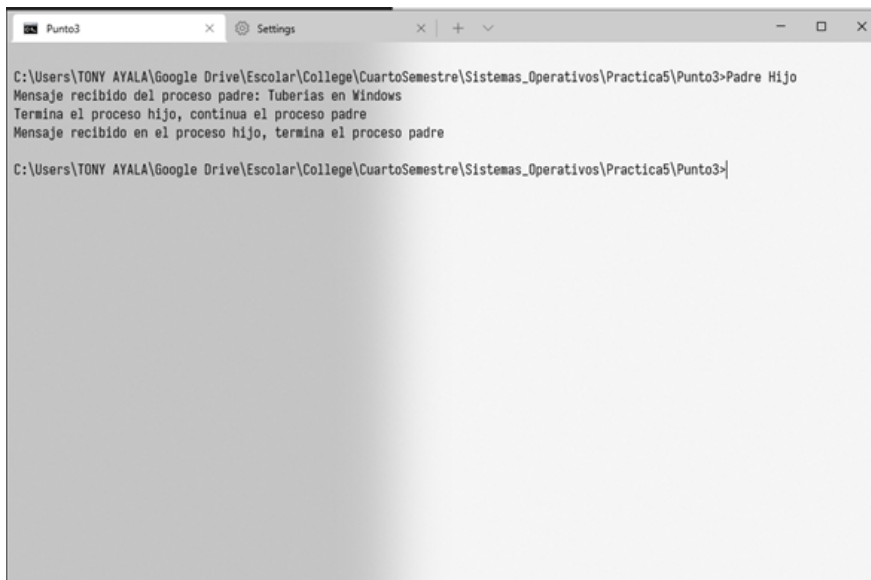


```

/* Programa hijo.c */
#include "windows.h"
#include "stdio.h"
int main ()
{
char mensaje[20];
DWORD leidos;
HANDLE hStdIn = GetStdHandle(STD_INPUT_HANDLE);
SECURITY_ATTRIBUTES pipeSeg =
{sizeof(SECURITY_ATTRIBUTES), NULL, TRUE};
/* Lectura desde la tubería sin nombre */
ReadFile(hStdIn, mensaje, sizeof(mensaje), &leidos, NULL);
printf("Mensaje recibido del proceso padre: %s\n", mensaje);
CloseHandle(hStdIn);
printf("Termina el proceso hijo, continua el proceso padre\n");
return 0;
}

```

Compilación



```

C:\Users\TONY AYALA\Google Drive\Escolar\College\CuartoSemestre\Sistemas_Operativos\Practica5\Punto3>Padre Hijo
Mensaje recibido del proceso padre: Tuberías en Windows
Termina el proceso hijo, continua el proceso padre
Mensaje recibido en el proceso hijo, termina el proceso padre
C:\Users\TONY AYALA\Google Drive\Escolar\College\CuartoSemestre\Sistemas_Operativos\Practica5\Punto3>

```

El programa tiene un funcionamiento correcto, como podemos observar, el proceso hijo es el primero en ser ejecutado y muestra el mensaje que hemos enviado mediante el programa principal, posteriormente se da la notificación del momento en que ha concluido el proceso para así saber en que momento se está ejecutando el programa principal.

4. Programe 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.

LINUX

```
home > tony > P4.c > Q determinant(float [10][10], float)
1  #include <math.h>
2  #include <stdio.h>
3  #include <unistd.h>
4  #include <string.h>
5  #include <stdlib.h>
6
7  float determinant (float a[10][10], float k){
8      float s=1, det=0, b[10][10];
9      int i,j,m,n,c;
10     if (k==1){
11         return (a[0][0]);
12     }
13     else{
14         det=0;
15         for (c = 0; c < k; c++)
16         {
17             m=0;
18             n=0;
19             for (i = 0; i < k; i++)
20             {
21                 for (j = 0; j < k; j++)
22                 {
23                     b[i][j]=0;
24                     if (i!=0 && j!= c)
25                     {
26                         b[m][n]=a[i][j];
27                         if(n<(k-2))
28                             n++;
29                     }
30                     else
31                     {
32                         n=0;
33                         m++;
34                     }
35                 }
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 void transpose(float num[10][10],float fac[10][10],float r, FILE* archivo){
48     int i,j;
49     r=10;
50     float b[10][10], inverse[10][10],d;
51
52     for (i = 0; i < r; i++)
53     {
54         for (j = 0; j < r; j++)
55         {
56             b[i][j]=fac[j][i];
57         }
58     }
```

```

home > tony > C P4.c > determinat(float [10][10], float)
56     {
57         b[i][j]=fac[j][i];
58     }
59 }
60 d=determinant(num,r);
61 for (i = 0; i < r; i++)
62 {
63     for (j = 0; j < r; j++)
64     {
65         inverse[i][j]=b[i][j]/d;
66     }
67 }
68 }
69
70 fprintf(archivo, "La inversa de la matriz es: \n");
71
72 for (i = 0; i < r; i++)
73 {
74     for (j = 0; j < r; j++)
75     {
76         fprintf(archivo,"%t%f",inverse[i][j]);
77     }
78     fprintf(archivo,"\n");
79 }
80 }
81
82 }
83
84 void cofactor(float num[10][10],float f, FILE* archivo){
85     float b[10][10],fac[10][10];
86     int p,q,m,n,i,j;
87     for (q = 0; q < f; q++)
88     {
89         for (p = 0; p < f; p++)
90         {
91             m=0;
92             n=0;
93             for (i = 0; i < f; i++)
94             {
95                 for (j = 0; j < f; j++)
96                 {
97                     if (i!= q && j!= p)
98                     {
99                         b[m][n]= num[i][j];
100                         if (n<(f-2))
101                             n++;
102                         else{
103                             n=0;
104                             m++;
105                         }
106                     }
107                 }
108             }
109         }
110     }
111 }
112 fac[q][p] = pow(-1, q + p)* determinant(b,f-1);

```

```

119 void conversionMat(int origi[][10], float transf[][10]){
120     for (int i = 0; i < 10; i++)
121     {
122         for (int j = 0; j < 10; j++)
123         {
124             transf[i][j] = (float) origi[i][j];
125         }
126     }
127 }
128
129 }
130
131
132 void sumMatriz(int matrices[][10][10],int result[][10]){
133     for (int i = 0; i < 10; i++)
134     {
135         for (int j = 0; j < 10; j++)
136         {
137             result[i][j] = matrices[0][i][j] + matrices[1][i][j];
138         }
139     }
140 }
141
142 }
143
144 void multMatriz(int matrices[][10][10],int result[][10]){
145     for (int k = 0; k < 10; k++)
146     {
147         for (int i = 0; i < 10; i++)
148         {
149             int elemento = 0;
150             for (int j = 0; j < 10; j++)
151             {
152                 elemento+= (matrices[0][k][j] * matrices [1][j][i]);
153             }
154             result[k][i]=elemento;
155         }
156     }
157 }
158
159 }
160 }
161
162 void llenaMatriz (int arreglo[][10][10],int numMatrices){
163     for (int i = 0; i < numMatrices; i++)
164     {
165         printf("Introduce los valores de la Matriz: %d \n", (i+1));
166         for (int j = 0; j < 10; j++)
167         {
168             for (int k = 0; k < 10; k++)
169             {
170                 scanf("%d", &arreglo[i][j][k]);
171             }
172             printf("\n");
173         }
174         printf("\n");
175     }
176 }

```

```

178 void imprimeMatriz (int arreglo[][10],int numMatrices
179     for (int i = 0; i < numMatrices; i++)
180     {
181         printf("Matriz %d: \n", (i+1));
182         for (int j = 0; j < 10; j++)
183         {
184             for (int k = 0; k < 10; k++)
185             {
186                 printf("%d, ", arreglo[i][j][k]);
187             }
188             printf("\n");
189         }
190         printf("\n");
191     }
192 }
193
194
195 void imprimeMatrizuno (int arreglo[][10]){
196
197     for (int j = 0; j < 10; j++)
198     {
199         for (int k = 0; k < 10; k++)
200         {
201             printf("%d, ", arreglo[j][k]);
202         }
203         printf("\n");
204     }
205     printf("\n");
206 }
207
208 void imprimeMatrizflo (float arreglo[][10]){
209
210     for (int j = 0; j < 10; j++)
211     {
212         for (int k = 0; k < 10; k++)
213         {
214             printf("%f, ", arreglo[j][k]);
215         }
216         printf("\n");
217     }
218     printf("\n");
219 }
220
221
222 int main(){
223
224     pid_t pidHijo, pidNieta;
225     int matrices1 [2][10][10];
226     printf("Ingresa matrices a multiplicar");
227     llenaMatriz(matrices1,2);
228
229     int fds1[2],fds2[2],fds3[2],fds4[2];
230     int buff[2][10][10];
231     int buff3[10][10];
232     int buff4[10][10];
233
234     if (pipe(fds1) !=0) exit(1);

```

```

246     arrMatriz2[0][0]=1;
247     if (pipe(fds2) !=0) exit(1);
248
249
250     printf("Ingresa matrices a Sumar \n");
251     llenaMatriz(matrices2,2);
252     if (pidNieto=Fork())==0{
253         close(fds2[1]);
254         read(fds2[0],buff2,sizeof(buff2));
255         int sumResul[10][10];
256         sumMatriz(buff2,sumResul);
257
258         close(fds4[0]);
259         write(fds4[1],sumResul,sizeof(sumResul));
260
261     }else{
262         close(fds2[0]);
263         write(fds2[1],matrices2,sizeof(matrices2));
264
265         close(fds3[0]);
266         write(fds3[1],multResul,sizeof(multResul));
267
268         wait(0);
269
270     }
271
272     close(fds3[0]);
273     write(fds3[1],multResul,sizeof(multResul));
274 }else{
275     close(fds1[0]);
276     write(fds1[1],matrices1,sizeof(matrices1));
277
278     close(fds3[1]);
279     read(fds3[0],buff3,sizeof(buff3));
280     printf("Del hijo llego la siguiente matriz (multi): \n");
281     imprimeMatrizuno(buff3);
282
283     close(fds4[1]);
284     read(fds4[0],buff4,sizeof(buff4));
285
286     printf("Del nieto llego la siguiente matriz (suma): \n");
287     imprimeMatrizuno(buff4);
288
289     wait(0);
290
291     float inversaUno [10][10];
292     conversionMat(buff3,inversaUno);
293
294     FILE* archivo = fopen("salida1.txt","w");
295     float d1= determinant(inversaUno,10);
296     if (d1!=0)
297         cofactor(inversaUno,10,archivo);

```

Del hijo llego la siguiente matriz (multi):

```
822, 1428, 1706, 1378, 718, 436, 1154, 1428, 1706, 1852,
822, 1428, 1706, 1378, 718, 436, 1154, 1428, 1706, 1852,
822, 1428, 1706, 1378, 718, 436, 1154, 1428, 1706, 1852,
822, 1428, 1706, 1378, 718, 436, 1154, 1428, 1706, 1852,
1416, 2352, 2696, 2104, 1048, 832, 1880, 2352, 2696, 2512,
828, 1432, 1704, 1372, 712, 444, 1156, 1432, 1704, 1646,
828, 1432, 1704, 1372, 712, 444, 1156, 1432, 1704, 1646,
828, 1432, 1704, 1372, 712, 444, 1156, 1432, 1704, 1646,
828, 1432, 1704, 1372, 712, 444, 1156, 1432, 1704, 1646,
828, 1432, 1704, 1372, 712, 444, 1156, 1432, 1704, 1646,
```

Del nieto llego la siguiente matriz (suma):

```
12, 24, 32, 28, 16, 4, 20, 24, 32, 28,
12, 24, 32, 28, 16, 4, 20, 24, 32, 28,
12, 24, 32, 28, 16, 4, 20, 24, 32, 28,
12, 24, 32, 28, 16, 4, 20, 24, 32, 28,
12, 24, 32, 28, 16, 4, 20, 24, 32, 160,
18, 28, 30, 22, 10, 12, 22, 28, 30, 20,
18, 28, 30, 22, 10, 12, 22, 28, 30, 20,
18, 28, 30, 22, 10, 12, 22, 28, 30, 20,
18, 28, 30, 22, 10, 12, 22, 28, 30, 20,
18, 28, 30, 22, 10, 12, 22, 28, 30, 20,
```

Wikipedia:Matrices

Matrix A:

822	1428	1706	1378	718	436	1154	1428	1706	1852
822	1428	1706	1378	718	436	1154	1428	1706	1852
822	1428	1706	1378	718	436	1154	1428	1706	1852
822	1428	1706	1378	718	436	1154	1428	1706	1852
1416	2352	2696	2104	1048	832	1880	2352	2696	2512
828	1432	1704	1372	712	444	1156	1432	1704	1646
828	1432	1704	1372	712	444	1156	1432	1704	1646
828	1432	1704	1372	712	444	1156	1432	1704	1646
828	1432	1704	1372	712	444	1156	1432	1704	1646
828	1432	1704	1372	712	444	1156	1432	1704	1646

The determinant is 0, the matrix is not invertible.

OK

Cholesky decompositi...

☒ Display decimals, number of significant digits: 3

$$\begin{pmatrix} 2 & 2 & 2 & 3 & 2 & 3 & 2 & 3 & 4 & 3 \\ 2 & 2 & 2 & 2 & 3 & 3 & 4 & 6 & 7 & 7 \\ 2 & 4 & 3 & 6 & 8 & 8 & 6 & 7 & 6 & 3 \\ 4 & 7 & 5 & 8 & 9 & 10 & 3 & 2 & 7 & 4 \\ 4 & 4 & 4 & 3 & 3 & 3 & 2 & 2 & 3 & 3 \\ 3 & 3 & 3 & 4 & 4 & 6 & 5 & 8 & 9 & 7 \\ 3 & 4 & 7 & 5 & 7 & 9 & 8 & 10 & 2 & 4 \\ 3 & 6 & 9 & 10 & 5 & 3 & 6 & 9 & 5 & 2 \\ 2 & 2 & 3 & 3 & 4 & 7 & 10 & 9 & 6 & 7 \\ 4 & 7 & 9 & 8 & 5 & 2 & 6 & 7 & 9 & 8 \end{pmatrix}^{(-1)} = \begin{pmatrix} 2.6 & -1.4 & 1.1 & -1.0 & 0.12 & -0.19 & 0.28 & -1.1 & -0.38 & 0.91 \\ -7.2 & 4.8 & -3.0 & 2.7 & 1.2 & -0.077 & -1.2 & 3.6 & 1.3 & -3.5 \\ 2.8 & -3.0 & 1.4 & -1.3 & -0.63 & 0.70 & 0.73 & -1.9 & -0.69 & 2.1 \\ 0.80 & 0.67 & -0.18 & 0.16 & -0.16 & -0.72 & -0.085 & 0.32 & 0.098 & -0.33 \\ 2.2 & -1.4 & 1.3 & -0.96 & -0.29 & -0.17 & 0.43 & -1.3 & -0.55 & 1.2 \\ -0.55 & 0.11 & -0.40 & 0.31 & -0.044 & 0.26 & -0.0047 & 0.22 & 0.12 & -0.27 \\ 0.51 & -0.70 & 0.40 & -0.31 & 0.015 & 0.0039 & -0.0097 & -0.38 & 0.11 & 0.42 \\ -1.8 & 1.5 & -0.79 & 0.62 & 0.30 & -0.046 & -0.25 & 1.0 & 0.22 & -1.1 \\ 1.3 & -2.0 & 0.99 & -0.81 & -0.27 & 0.74 & 0.24 & -1.2 & -0.37 & 1.2 \\ -0.54 & 1.5 & -0.72 & 0.58 & 0.0025 & -0.66 & -0.12 & 0.67 & 0.23 & -0.73 \end{pmatrix}$$

Clean +
Insert in A
Insert in B
Clean

The screenshot shows a software interface with a search bar at the top. Below it, there are two matrices, `salda1.txt` and `salda2.txt`, which are being multiplied. The result is displayed as a 10x10 matrix. The interface includes a file explorer on the left, a search bar at the top, and a settings menu on the right.

	salda1.txt	×	salda2.txt							
1	2.6	-1.4	1.1	-1.0	0.12	-0.19	0.28	-1.1	-0.38	0.91
2	-7.2	4.8	-3.0	2.7	1.2	-0.07	-1.2	3.6	1.3	-3.5
3	32.8	-3.8	1.4	-1.3	-0.63	0.70	0.73	-1.9	-0.69	2.1
4	0.80	0.67	-0.18	0.16	-0.16	-0.72	-0.08	0.32	0.09	-0.33
5	52.2	-1.4	1.3	-0.96	-0.29	-0.17	0.43	-1.31	-0.55	1.2
6	-0.55	0.11	-0.40	0.31	-0.04	0.26	-0.00	0.22	0.12	-0.27
7	0.51	-0.70	0.40	-0.31	0.01	0.00	-0.00	-0.38	0.11	0.42
8	-1.8	1.5	-0.79	0.62	0.30	-0.04	-0.25	1.00	0.22	-1.1
9	1.30	-2.00	0.99	-0.81	-0.27	0.74	0.24	-1.20	-0.37	1.22
10	-0.52	1.51	-0.72	0.58	0.00	-0.66	-0.12	0.67	0.23	-0.73


```
+  ×  salida1.txt  salida2.txt  ↺
1 La matriz no tiene inversa
```

Sección Windows

Padre.c

```
#include <stdio.h>
#include <math.h>
#include <windows.h>

float determinant(float a[10][10], float b){
    float s=1, det=0, b[10][10];
    int i,j,m,n,c;
    if(b==1){
        return (a[0][0]);
    }
    else
    {
        det=0;
        for(c=0; c<b; c++)
        {
            m=0;
            n=0;
            for (int i = 0; i < b; i++)
            {
                for (int j = 0; j < b; j++)
                {
                    b[i][j]=0;
                    if (i!=0 && j!=c)
                    {
                        b[m][n]=a[i][j];
                        if (n<(b-2))
                        {
                            n++;
                        }
                        else{
                            n=0;
                            m++;
                        }
                    }
                }
            }
            det+=s*(a[0][c]*determinant(b,b-1));
            s= -1*s;
        }
        return (det);
    }
}

void transpose(float num[10][10], float fac[10][10], float r, FILE* archivo){
    int i,j;
    r=10;
    float b[10][10], inverse[10][10], d;

    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            b[i][j]=fac[j][i];
        }
    }
    d=determinant(num,r);
    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            inverse[i][j]=b[i][j]/d;
        }
    }

    fprintf(archivo, "La inversa de la matriz es: \n");

    for (i = 0; i < r; i++)
    {
        for (j = 0; j < r; j++)
        {
            fprintf(archivo, "%t\t", inverse[i][j]);
        }
        fprintf(archivo, "\n");
    }
}

void cofactor(float num[10][10], float f, FILE* archivo){
    float b[10][10], fac[10][10];
    int p,q,m,n,i,j;
    for (q = 0; q < f; q++)
    {
        for (p = 0; p < f; p++)
        {
            /* code */
            m=0;
            n=0;
            for (i = 0; i < f; i++)
```

```

        m=0;
        n=0;
        for (i = 0; i < f; i++)
        {
            for (j = 0; j < f; j++)
            {
                if (i!=q && j!=0)
                {
                    b[m][n]= num[i][j];
                    if (n<(f-2))
                        n++;
                    else{
                        n=0;
                        m++;
                    }
                }
            }
        }

        fac[q][p] = pow(-1,q+p)* determinant(b,f-1);
    }

    transpose(num,fac,f,archivo);
}

int main (){
    //creando tuberia
    return 0;
    SECURITY_ATTRIBUTES saAttr;
    saAttr.nLength=sizeof(SECURITY_ATTRIBUTES);
    saAttr.bInheritHandle=TRUE;
    saAttr.lpSecurityDescriptor=NULL;

    HANDLE hRead1, hWrite1, hRead2, hWrite2;

    CreatePipe(&hRead1,&hWrite1,&saAttr,0);
    CreatePipe(&hRead2,&hWrite2,&saAttr,0);

    //Mandando Mensaje
    int i,j;
    float szBuffer[10][20];
    for (i = 0; i < 10; i++)
        for (j = 0; j < 20; j++)
            szBuffer[i][j]=rand() % 10;

    DWORD dwBufferSize = sizeof(szBuffer);
    DWORD dwNoBytesWrite;

    BOOL bWriteFile = WriteFile(hWrite1,szBuffer,dwBufferSize,&dwNoBytesWrite,NULL);

    //Creando Hijo
    STARTUPINFO si1;
    GetStartupInfo(&si1);
    PROCESS_INFORMATION pi1;
    ZeroMemory (&si1, sizeof(si1));
    ZeroMemory (&pi1, sizeof(pi1));
    si1.cb = sizeof(STARTUPINFO);
    si1.hStdError = GetStdHandle (STD_ERROR_HANDLE);
    si1.hStdOutput = hWrite2;
    si1.hStdInput= hRead1;
    si1.dwFlags = STARTF_USESTDHANDLES;

    printf("Padre : %f\n",szBuffer[0][0]);
    BOOL process1 = CreateProcess("Hijo.exe",NULL, NULL, NULL, TRUE, 0, NULL, NULL, &si1, &pi1);
    if(process1 ==FALSE) printf("NO");

    WaitForSingleObject(pi1.hProcess,INFINITE);

    CloseHandle(pi1.hProcess);
    CloseHandle(pi1.hThread);

    //Recibiendo Mensaje
    DWORD dwNoBytesRead;
    printf("Padre: %f\n",szBuffer[0][0]);
    BOOL bReadFile = ReadFile(hRead2,szBuffer,dwBufferSize,&dwNoBytesRead,NULL);
    printf("Padre: %f\n",szBuffer[0][0]);

    CloseHandle(hRead1);
    CloseHandle(hWrite1);
    CloseHandle(hRead2);
    CloseHandle(hWrite2);
}

```

```

printf("Padre : %f\n",szBuffer[0][0]);
BOOL process1 = CreateProcess("Hijo.exe",NULL, NULL, NULL, TRUE, 0, NULL, NULL, &si1, &pi1);
if(process1 ==FALSE) printf("NO");

WaitForSingleObject(pi1.hProcess,INFINITE);

CloseHandle(pi1.hProcess);
CloseHandle(pi1.hThread);

//Recibiendo Mensaje
DWORD dwNoBytesRead;
printf("Padre: %f\n",szBuffer[0][0]);
BOOL bReadFile = ReadFile(hRead2,szBuffer,dwBufferSize,&dwNoBytesRead,NULL);
printf("Padre: %f\n",szBuffer[0][0]);

CloseHandle(hRead1);
CloseHandle(hWrite1);
CloseHandle(hRead2);
CloseHandle(hWrite2);
//=====

//Escribiendo Resultados
float a[10][10], b[10][10];
for (i = 0; i < 10; i++)
{
    for (j = 0; j < 20; j++)
    {
        if(j<10)
        {
            a[i][j]= szBuffer[i][j];
        }
        else
        {
            b[i][j-10]= szBuffer[i][j];
        }
    }
}

FILE* archivo = fopen("salida1.txt","w");
float d1 = determinant(a,10);

if (d1!=0)
{
    cofactor(a,10,archivo);
}
else
{
    fprintf(archivo,"La matriz no tiene inversa");
}
fclose(archivo);

archivo= fopen("salida2.txt","w");

float d2 = determinant(b,10);

if (d2!=0)
{
    cofactor(b,10,archivo);
}
else
{
    fprintf(archivo,"La matriz no tiene inversa");
}
fclose(archivo);

return 0;

```

Hijo.c

```
#include <stdio.h>
#include <stdlib.h>

int main(){

    // Recepción de Datos

    HANDLE hstdin = GetStdHandle(STD_INPUT_HANDLE);
    HANDLE hstdout = GetStdHandle(STD_OUTPUT_HANDLE);

    int szbuffer[10][10];
    DWORD dbufferSize = sizeof(szbuffer);
    DWORD dbufferSize2;
    BOOL hwriteFile;
    BOOL hreadFile;

    printf("Mija: %d\n", szbuffer[0][0]);
    hreadFile = ReadFile(hstdin, szbuffer, dbufferSize, &dbufferSize2, 0);
    CloseHandle(hstdin);

    // Datos

    // Preparación de Datos
    int i, j;
    float a[10][10], b[10][10];

    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < 10; j++)
        {
            if (j < 10)
            {
                a[i][j] = szbuffer[i][j];
            }
            else
            {
                b[i][j-10] = szbuffer[i][j];
            }
        }
    }

    // Mando los datos al hijo
    SECURITY_ATTRIBUTES saAttr;
    saAttr.nLength = sizeof(SECURITY_ATTRIBUTES);
    saAttr.bInheritable = TRUE;
    saAttr.lpSecurityDescriptor = NULL;

    HANDLE hread1, hwrite1, hread2, hwrite2;

    CreatePipe(&hread1, &hwrite1, &saAttr, 0);
    CreatePipe(&hread2, &hwrite2, &saAttr, 0);

    hwriteFile = WriteFile(hwrite1, szbuffer, dbufferSize, &dbufferSize2, 0);

    // Creando hijo (nieto)
    STARTUPINFO si;
    GetStartupInfo(&si);
    PROCESS_INFORMATION pi;
    ZeroMemory(&pi, sizeof(pi));
    ZeroMemory(&si, sizeof(si));
    si.cb = sizeof(STARTUPINFO);
    si.hStdError = GetStdHandle(STD_ERROR_HANDLE);
    si.hStdOutput = hwrite2;
    si.hStdInput = hread1;
    si.dwFlags = STARTF_USESTDHANDLES;

    BOOL process = CreateProcess("C:\\Users\\TONY AYALA\\Google Drive\\Escuela\\College\\Cuatrimestre\\Sistemas Operativos\\Practical\\programas\\Nieto.exe", NULL, NULL, NULL, TRUE, 0, NULL, &si, &pi);
    WaitForSingleObject(pi.hProcess, INFINITE);

    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);

    // Recibiendo mensaje del hijo (nieto)
    hreadFile = ReadFile(hread2, szbuffer, dbufferSize, &dbufferSize2, 0);

    CloseHandle(hread1);
    CloseHandle(hwrite1);
    CloseHandle(hread2);
    CloseHandle(hwrite2);

    // Mando datos al padre
    printf("Multiplicacion: \n");
    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < 10; j++)
        {
            szbuffer[i][j] = a[i][j] * b[i][j];
            printf("%f", szbuffer[i][j]);
        }
    }
}
```

Nieto.c

```
#include<windows.h>
#include<stdio.h>

int main(){
    HANDLE hStdin = GetStdHandle (STD_INPUT_HANDLE);
    HANDLE hStdout = GetStdHandle(STD_OUTPUT_HANDLE);

    int szBuffer[10][20];
    DWORD dwBufferSize = sizeof(szBuffer);
    DWORD dwNoBytesRead;
    DWORD dwNoBytesWrite;
    BOOL bReadFile;
    bReadFile = ReadFile(hStdin,szBuffer,dwBufferSize,&dwNoBytesRead,NULL);
    CloseHandle(hStdin);

    int i,j;
    float a[10][10],b[10][10];

    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < 20; j++)
        {
            if (j<10)
            {
                a[i][j]=szBuffer[i][j];
            }
            else
            {
                b[i][j-10]=szBuffer[i][j];
            }
        }
    }

    printf("Suma: \n");
    for (i = 0; i < 10; i++)
    {
        for (j = 0; j < 10; j++)
        {
            szBuffer[i+10][j+10]=a[i][j]+b[i][j];
            printf("%.2f",szBuffer[i+10][j+10]);
        }
        printf("\n");
    }

    BOOL bWriteFile=WriteFile(hStdout,szBuffer,dwBufferSize,&dwNoBytesWrite,NULL);
    CloseHandle(hStdout);

    return 0;
}
```

236	197	274	219	217	233	171	279	160	246
201	172	221	158	175	185	139	236	144	211
197	156	237	172	191	187	133	219	143	214
205	181	286	194	207	240	198	242	145	222
189	147	258	172	211	212	164	250	164	249
270	213	329	203	230	288	200	295	187	252
270	248	331	249	248	302	214	289	200	277
209	175	288	172	196	261	169	244	178	229
242	199	316	214	222	285	195	252	214	281
310	237	349	264	253	342	239	319	247	312

11	10	6	6	14	5	9	9	8	14
7	8	9	4	6	5	2	6	11	8
9	12	9	6	4	11	9	12	10	5
12	11	15	13	5	9	14	9	5	10
9	4	11	11	11	9	13	13	7	14
13	11	13	11	7	10	10	18	9	11
12	16	13	13	6	13	8	7	12	11
6	8	8	16	12	5	8	14	3	9
16	4	10	9	7	16	7	9	7	7
6	7	10	11	12	10	9	15	6	58

```

Resultado1 - Notepad
File Edit Format View Help
Resultado de la inversa de la Suma
0.0103 -0.0182 -0.0229 -0.0251 0.0165 0.0590 0.0052 -0.0620 0.0305 -0.0097
-0.0128 0.0235 -0.0026 0.0022 -0.0088 -0.0126 0.0178 0.0092 -0.0023 -0.0088
-0.0048 0.0195 0.0505 0.0427 -0.0506 -0.0399 -0.0465 0.0511 -0.0030 0.0047
0.0093 0.0022 0.0423 0.0287 -0.0358 -0.0531 -0.0248 0.0437 -0.0224 0.0234
-0.0484 -0.0166 0.0357 -0.0126 0.0311 0.0029 0.0439 -0.0030 -0.0469 0.0121
0.0512 -0.0578 -0.1034 -0.0733 0.0664 0.0903 0.0540 -0.0756 0.0495 -0.0411
-0.0333 0.0248 0.0174 0.0385 -0.0101 -0.0252 -0.0134 0.0070 -0.0208 0.0239
0.0034 0.0193 0.0154 0.0170 -0.0160 -0.0341 -0.0217 0.0466 -0.0319 0.0158
-0.0910 0.0668 0.1468 0.0794 -0.0780 -0.1413 -0.0476 0.1412 -0.1056 0.0795
0.0741 -0.0382 -0.1285 -0.0680 0.0651 0.1053 0.0297 -0.1189 0.1097 -0.0678
Resultado de la inversa de la Multiplicacion
-0.0000 -0.0000 0.0000 -0.0000 -0.0000 -0.0000 -0.0000 0.0000 0.0000 -0.0000
0.0549 0.0028 0.0620 0.0641 -0.1437 -0.0251 -0.0350 0.0351 0.0116 0.0092
0.0204 0.1323 -0.0902 0.1517 -0.1214 0.0428 -0.1243 -0.0171 0.0791 -0.0037
-0.0719 -0.0757 -0.0682 -0.1013 0.1269 0.0272 0.1629 0.0259 -0.0721 -0.0109
0.1049 0.0388 -0.0551 0.0357 -0.0760 -0.0109 -0.0663 0.0209 0.0771 -0.0116
0.0018 -0.0073 0.0773 0.0203 -0.0730 -0.0729 -0.0455 0.0230 0.1073 0.0140
0.0133 -0.0940 0.0329 -0.0082 0.1343 -0.0163 0.0511 -0.0601 -0.0730 -0.0126
0.0089 -0.0470 -0.0361 -0.0526 -0.0089 0.1487 -0.0050 -0.0194 -0.0087 -0.0045
-0.0506 -0.0320 -0.0450 -0.1681 0.2257 0.0554 0.1808 -0.0769 -0.1184 -0.0236
-0.0162 0.0121 0.0242 0.0053 0.0012 -0.0322 -0.0066 0.0057 -0.0102 0.0239
Ln 1, Col 1 100% Windows (CRLF) UTF-8

```

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

```
#include <sys/types.h> /* Cliente de la memoria compartida */
#include <sys/ipc.h>
#include <sys/shm.h>
#include <stdio.h>
#define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
int main()
{
    int shmid;
    key_t llave;
    char *shm, *s;
    llave = 5678;
    if ((shmid = shmget(llave, TAM_MEM, 0666)) < 0) {
        perror("Error al obtener memoria compartida: shmget");
        exit(-1);
    }
    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
        perror("Error al enlazar la memoria compartida: shmat");
        exit(-1);
    }
    for (s = shm; *s != '\0'; s++)
        putchar(*s);
    putchar('\n');
    *shm = '*';
    exit(0);
}
```



```
#include <sys/types.h> /* Servidor de la memoria compartida */
#include <sys/ipc.h> /* (ejecutar el servidor antes de ejecutar el
cliente)*/
#include <sys/shm.h>
#include <stdio.h>
#define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
int main()
{
    char c;
    int shmid;
    key_t llave;
    char *shm, *s;
    llave = 5678;
    if ((shmid = shmget(llave, TAM_MEM, IPC_CREAT | 0666)) < 0) {
        perror("Error al obtener memoria compartida: shmget");
        exit(-1);
    }
    if ((shm = shmat(shmid, NULL, 0)) == (char *) -1) {
        perror("Error al enlazar la memoria compartida: shmat");
        exit(-1);
    }
    s = shm;
    for (c = 'a'; c <= 'z'; c++)
        *s++ = c;
    *s = '\0';
    while (*shm != '*')
        sleep(1);
    exit(0);
}
```

Ejecución
Servidor

```
jomiantc@jomiantc-SVF15215CLW: ~/Escritorio
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$ gcc servidor.c -o s.exe
servidor.c: In function 'main':
servidor.c:27:2: warning: implicit declaration of function 'sleep' [-Wimplicit-f
unction-declaration]
   27 |     sleep(1);
      |     ~~~~~
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$ ./s.exe
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$
```

Cliente

```
jomiantc@jomiantc-SVF15215CLW: ~/Escritorio
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$ gcc cliente.c -o c.exe
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$ ./c.exe
abcdefghijklmnopqrstuvwxyz
jomiantc@jomiantc-SVF15215CLW:~/Escritorio$
```

El programa funciona de manera optima, el servidor crea el espacio de memoria compartida que guarda en dicho espacio de memoria las letras del abecedario una vez que lo ejecutamos el servidor espera a que otro programa acceda al espacio para leer la información guardarla en un arreglo de caracteres e imprimirlo en la consola del cliente

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

```
#include <windows.h> /* Cliente de la memoria compartida */
#include <stdio.h>
#define TAM_MEM 27 /*Tamaño de La memoria compartida en bytes */
int main(void)
{
    HANDLE hArchMapeo;
    char *idMemCompartida = "MemoriaCompatida";
    char *apDatos, *apTrabajo, c;
    if((hArchMapeo=OpenFileMapping(
    FILE_MAP_ALL_ACCESS, // acceso lectura/escritura de La memoria
    compartida
```

```

FALSE, // no se hereda el nombre
idMemCompartida) // identificador de la memoria compartida
) == NULL)
{
printf("No se abrio archivo de mapeo de la memoria compartida:
(%i)\n", GetLastError());
exit(-1);
}
if((apDatos=(char *)MapViewOfFile(hArchMapeo, // Manejador del mapeo
FILE_MAP_ALL_ACCESS, // Permiso de lectura/escritura en la memoria
0,
0,
TAM_MEM)) == NULL)
{
printf("No se accedio a la memoria compartida: (%i)\n",
GetLastError());
CloseHandle(hArchMapeo);
exit(-1);
}

for (apTrabajo = apDatos; *apTrabajo != '\0'; apTrabajo++)
putchar(*apTrabajo);
putchar('\n');
*apDatos = '*';
UnmapViewOfFile(apDatos);
CloseHandle(hArchMapeo);
exit(0);
}

```

```

#include <windows.h> /* Servidor de la memoria compartida */
#include <stdio.h> /* (ejecutar el servidor antes de ejecutar el
cliente)*/
#define TAM_MEM 27 /*Tamaño de la memoria compartida en bytes */
int main(void)
{
HANDLE hArchMapeo;

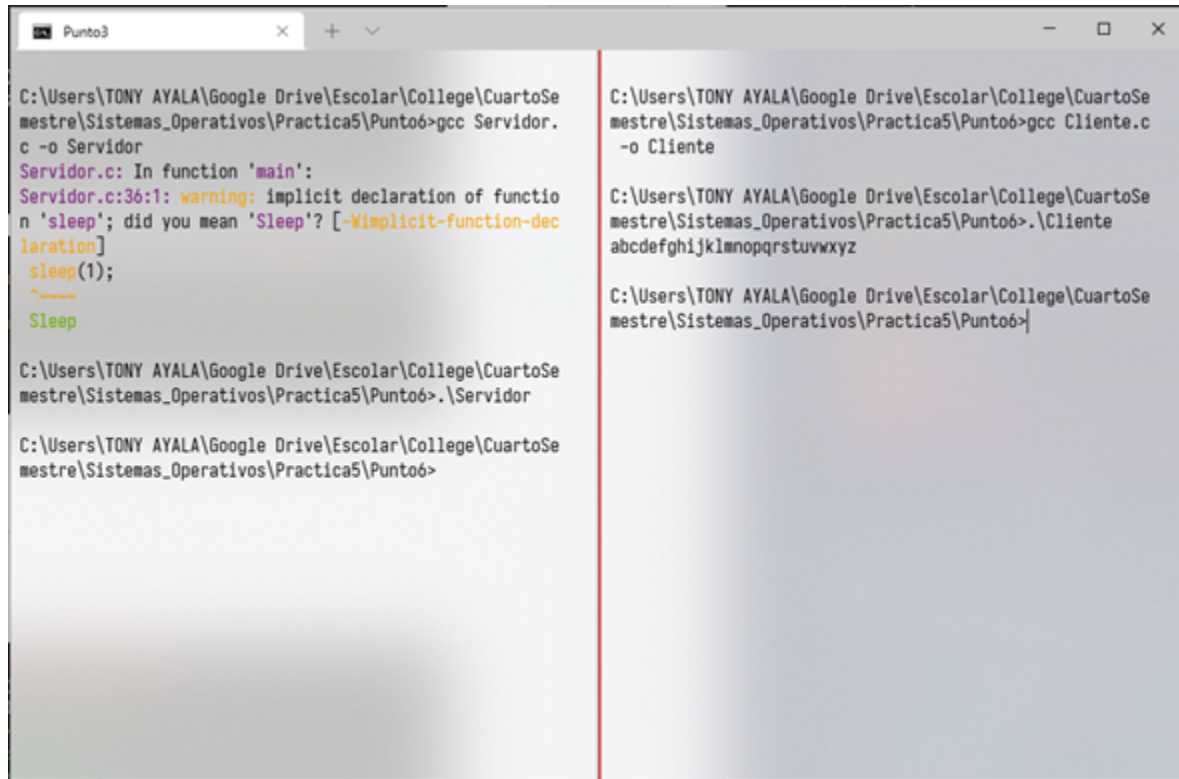
```

```

char *idMemCompartida = "MemoriaCompatida";
char *apDatos, *apTrabajo, c;
if((hArchMapeo=CreateFileMapping(
INVALID_HANDLE_VALUE, // usa memoria compartida
NULL, // seguridad por default
PAGE_READWRITE, // acceso lectura/escritura a la memoria
0, // tamaño maximo parte alta de un DWORD
TAM_MEM, // tamaño maximo parte baja de un DWORD
idMemCompartida) // identificador de la memoria compartida
) == NULL)
{
printf("No se mapeo la memoria compartida: (%i)\n", GetLastError());
exit(-1);
}
if((apDatos=(char *)MapViewOfFile(hArchMapeo, // Manejador del mapeo
FILE_MAP_ALL_ACCESS, // Permiso de lectura/escritura en la memoria
0,
0,
TAM_MEM)) == NULL)
{
printf("No se creo la memoria compartida: (%i)\n", GetLastError());
CloseHandle(hArchMapeo);
exit(-1);
}
apTrabajo = apDatos;
for (c = 'a'; c <= 'z'; c++)
*apTrabajo++ = c;
*apTrabajo = '\0';
while (*apDatos != '*')
sleep(1);
UnmapViewOfFile(apDatos);
CloseHandle(hArchMapeo);
exit(0);
}

```

Compilación



```
Punto3
C:\Users\TONY AYALA\Google Drive\Escolar\College\CuartoSemestre\Sistemas_Operativos\Practica5\Punto6>gcc Servidor.c -o Servidor
Servidor.c: In function 'main':
Servidor.c:36:1: warning: implicit declaration of function 'sleep'; did you mean 'Sleep'? [-Wimplicit-function-declaration]
sleep(1);
^~~~~~
Sleep

C:\Users\TONY AYALA\Google Drive\Escolar\College\CuartoSemestre\Sistemas_Operativos\Practica5\Punto6>.\Servidor

C:\Users\TONY AYALA\Google Drive\Escolar\College\CuartoSemestre\Sistemas_Operativos\Practica5\Punto6>

C:\Users\TONY AYALA\Google Drive\Escolar\College\CuartoSemestre\Sistemas_Operativos\Practica5\Punto6>gcc Cliente.c -o Cliente

C:\Users\TONY AYALA\Google Drive\Escolar\College\CuartoSemestre\Sistemas_Operativos\Practica5\Punto6>.\Cliente
abcdefghijklmnopqrstuvwxyz

C:\Users\TONY AYALA\Google Drive\Escolar\College\CuartoSemestre\Sistemas_Operativos\Practica5\Punto6>
```

Como podemos observar el programa en cuestión funciona de forma adecuada, vemos que el Servidor crea el espacio de memoria correspondiente para almacenar las letras del abecedario las cuales permanecen ahí hasta que otro programa intente acceder a esta memoria compartida, en este caso el Cliente al momento de ser ejecutado obtiene las letras del abecedario y las imprime en pantalla como debería ser

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

Código y Compilación

Programa Padre.c

```
1  #include <stdio.h>          //LIBRERIAS BASICAS PARA C
2  #include <stdlib.h>
3  #include <sys/types.h>      //LIBRERIAS PARA LA CERACION DE PROCESOS
4  #include <sys/wait.h>
5  #include <sys/ipc.h>        //LIBRERIAS PARA LA CREACION DE MEMORIA COMPARTIDA
6  #include <sys/shm.h>
7  #include <unistd.h>         //LIBRERIAS VARIAS PARA EL FUNCIONAMIENTO DEL PROGRAMA
8
9  #define TAM_MEM 27
10
11 void enviaMatrizAB ();
12 void reciboMatrizM ();
13 void reciboMatrizS ();
14 void inversa();
15
16 // ENVIO PARA LAS MATRICES A Y B
17
18 int shmidAB;
19 key_t llaveAB = 5678;
20 char *shmAB, *sAB;
21
22 // RECIBO PARA LA MATRIZ M
23
24 int shmidM;
25 key_t llaveM = 5680;
26 char *shmM, *sM;
27
28 // RECIBO PARA LA MATRIZ S
29
30 int shmidS;
31 key_t llaveS = 5684;
32 char *shmS, *sS;
33
34 // MATRICES
35
36 int matrizA[10][10] = { ...
48 };
49
50 int matrizB[10][10] = { ...
62 };
63
64 double identidadM[10][10] = { ...
76 };
77
78 double identidadS[10][10] = { ...
90 };
91
92 int matrizM[10][10];
93
94 int matrizS[10][10];
95
96 // VARIABLES VARIAS
97
98 int i = 0, j = 0, k = 0, aux1 = 0, aux2 = 0, aux3 = 0, aux4 = 0;
99
100 double auxS = 0, pivotes = 0, saveS[10][10];
101 double auxM = 0, pivoteM = 0, saveM[10][10];
```

```

103 int main(){
104     //CODIGO PARA LA CREACION DE PROCESOS
105
106     pid_t hijo;
107     char *argv[2];
108
109     argv[0]="/home/jomiantc/Escritorio/HIJO.exe";
110     argv[1]="/home/jomiantc/Escritorio/NIETO.exe";
111     argv[2]=NULL;
112
113     //CODIGO QUE SE EJECUTA ANTES DE CREAR EL HIJO
114
115     enviaMatrizAB();
116
117     if((hijo=fork())==-1)
118         printf("Error al crear el proceso hijo\n");
119
120     if(hijo==0){
121         execv(argv[0],argv);
122     }
123     else{
124
125         wait(0);
126         reciboMatrizM();
127         reciboMatrizS();
128         inversa();
129         exit(0);
130     }
131 }
132
133
134 void enviaMatrizAB (){
135     // CODIGO QUE ESTABLECE CONEXION PARA LAS MATRICES A Y B
136
137     if ((shmidAB = shmget(llaveAB, TAM_MEM, IPC_CREAT | 0666)) < 0) {
138
139         perror("Error al obtener memoria compartida P: shmget");
140         exit(-1);
141     }
142
143     if ((shmAB = shmat(shmidAB, NULL, 0)) == (char *) -1) {
144
145         perror("Error al enlazar la memoria compartida P: shmat");
146         exit(-1);
147     }
148
149     // CODIGO PARA ENVIAR MATRICES A Y B
150
151     SAB = shmAB;
152
153     for (i = 0; i < 10; i++){
154
155         for (j = 0; j < 10; j++){
156
157             *SAB++ = matrizA[i][j];
158         }
159     }
160
161
162     for (i = 0; i < 10; i++){
163
164         for (j = 0; j < 10; j++){
165
166             *SAB++ = matrizB[i][j];
167         }
168     }
169
170     *SAB++ = '\0';
171 }

```

```

173 void reciboMatrizM (){
174 // CODIGO QUE ESTABLECE LA CONEXION PARA LA MATRIZ M
175
176     if ((shmidM = shmget(llaveM, TAM_MEM, 0666)) < 0) {
177
178         perror("Error al obtener memoria compartida P: shmget");
179         exit(-1);
180     }
181
182     if ((shmm = shmat(shmidM, NULL, 0)) == (char *) -1) {
183
184         perror("Error al enlazar la memoria compartida P: shmat");
185         exit(-1);
186     }
187
188 //CODIGO QUE GUARDA LA MATRIZ M
189
190     for (sM = shmm; *sM != '\0'; sM++){
191
192         matrizM[aux1][aux2] = *sM;
193         aux2++;
194
195         if (aux2 == 10){
196
197             aux2 = 0;
198             aux1++;
199
200         }
201     }
202
203     *shmm = '*';
204 }
205
206 void reciboMatrizS (){
207 // CODIGO QUE ESTABLECE LA CONEXION PARA LA MATRIZ S
220
221 //CODIGO QUE GUARDA LA MATRIZ S
236 }

```



```

238 void inversa (){
239 // CODIGO QUE CALCULA LA INVERSA DE LAS MATRICES M Y S
240     for (i = 0; i < 10; i++){
241
242         for (j = 0; j < 10; j++){
243
244             saveM[i][j] = matrizM[i][j];
245             saveS[i][j] = matrizS[i][j];
246         }
247     }
248
249     for (i = 0; i < 10; i++){
250
251         pivoteM = saveM[i][i];
252         pivoteS = saveS[i][i];
253
254         for (k = 0; k < 10; k++){
255
256             saveM[i][k] = saveM[i][k]/pivoteM;
257             identidadM[i][k] = identidadM[i][k]/pivoteM;
258             saveS[i][k] = saveS[i][k]/pivoteS;
259             identidadS[i][k] = identidadS[i][k]/pivoteS;
260
261         }
262
263         for (j = 0; j < 10; j++){
264
265             if (i != j) {
266
267                 auxM = saveM[j][i];
268                 auxS = saveS[j][i];
269
270                 for (k = 0; k < 10; k++){
271
272                     saveM[j][k] = saveM[j][k] - auxM*saveM[i][k];
273                     identidadM[j][k] = identidadM[j][k] - auxM*identidadM[i][k];
274                     saveS[j][k] = saveS[j][k] - auxS*saveS[i][k];
275                     identidadS[j][k] = identidadS[j][k] - auxS*identidadS[i][k];
276
277                 }
278             }
279         }
280
281     FILE *fichero;
282     fichero = fopen("INVERSAS.txt","w");
283
284     fprintf(fichero, "Matriz inversa de la multiplicacion: \n");
285
286     for (i = 0; i < 10; i++){
287
288         for (j = 0; j < 10; j++){
289
290             fprintf(fichero, "%0.2lf, ", identidadM[i][j]);
291
292         }
293         fprintf(fichero, "\n");
294     }
295
296     fprintf(fichero, "\nMatriz inversa de la suma: \n");
297
298     for (i = 0; i < 10; i++){
299
300         for (j = 0; j < 10; j++){
301
302             fprintf(fichero, "%0.2lf, ", identidadS[i][j]);
303
304         }
305         fprintf(fichero, "\n");
306     }
307
308     fclose(fichero);
309 }

```

Código Hijo.c,

La mayor parte del código es similar al anterior por ello solo se mostrarán las funciones declaradas con un comentario que describe lo que hace

```
1  #include <stdio.h>      //LIBRERIAS BASICAS PARA C
2  #include <stdlib.h>
3  #include <sys/types.h>  //LIBRERIAS PARA LA CREACION DE PROCESOS
4  #include <sys/wait.h>
5  #include <sys/ipc.h>    //LIBRERIAS PARA LA CREACION DE MEMORIA COMPARTIDA
6  #include <sys/shm.h>
7  #include <unistd.h>     //LIBRERIAS VARIAS PARA EL FUNCIONAMIENTO DEL PROGRAMA
8
9  #define TAM_MEM 27
10
11 void reciboMatrizAB ();
12 void multiplicar ();
13 void enviarMatrizM ();
14 void enviarMatrizABS ();
15
16 // RECIBO PARA LAS MATRICES A Y B
17
18 int shmIdAB;
19 key_t llaveAB = 5678;
20 char *shmAB, *sAB;
21
22 // ENVIO PARA LA MATRIZ M
23
24 int shmIdM;
25 key_t llaveM = 5680;
26 char *shmM, *sM;
27
28 // ENVIO PARA LAS MATRICES AS Y BS
29
30 int shmIdABS;
31 key_t llaveABS = 5682;
32 char *shmABS, *sABS;
33
34 // MATRICES
35
36 int matrizA[10][10], matrizB[10][10], matrizM[10][10];
37
38 // VARIABLES VARIAS
39
40 int aux1 = 0, aux2 = 0, aux3 = 0, aux4 = 0, suma = 0, i = 0, j = 0, a = 0;
41
42 int main(int argc, char *argv[]){
43     //CODIGO PARA LA CREACION DE PROCESOS
44
45     pid_t nieto;
46
47     //CODIGO QUE SE EJECUTA ANTES DE CREAR EL NIETO
48
49     reciboMatrizAB();
50     multiplicar();
51     enviarMatrizM();
52     enviarMatrizABS();
53
54     if((nieto=fork())== -1)
55         printf("Error al crear el proceso nieto\n");
56
57     if(nieto==0){
58
59         execv(argv[1],argv);
60     }
61     else{
62
63         wait(0);
64         exit(0);
65     }
66 }
```

```
68 void reciboMatrizAB (){
69 ▶ // CODIGO QUE ESTABLECE LA CONEXION PARA LAS MATRICES A Y B ...
82
83 ▶ //CODIGO QUE GUARDA LAS MATRICES A Y B ...
114 }
115
116 void multiplicar (){
117 ▼ //CODIGO QUE MULTIPLICA LAS MATRICES A Y B
118
119 ▼ for (a = 0; a < 10; a++) {
120
121 ▼ for (i = 0; i < 10; i++) {
122
123 suma = 0;
124
125 ▼ for (j = 0; j < 10; j++) {
126
127 suma += matrizA[i][j] * matrizB[j][a];
128 }
129
130 matrizM[i][a] = suma+a;
131 }
132 }
133 }
134
135 void enviarMatrizM (){
136 ▶ // CODIGO QUE ESTABLECE CONEXION PARA LA MATRIZ M ...
149
150 ▶ // CODIGO PARA ENVIAR LA MATRIZ M ...
163 }
164
165 void enviarMatrizABS (){
166 ▶ // CODIGO QUE ESTABLECE CONEXION PARA LA MATRIZ S ...
179
180 ▶ // CODIGO PARA ENVIAR LA MATRIZ S ...
202 }
203
```

Código Nieto.c

```
1  #include <stdio.h>           //LIBRERIAS BASICAS PARA C
2  #include <stdlib.h>
3  #include <sys/types.h>       //LIBRERIAS PARA LA CERCACION DE PROCESOS
4  #include <sys/wait.h>
5  #include <sys/ipc.h>         //LIBRERIAS PARA LA CREACION DE MEMORIA COMPARTIDA
6  #include <sys/shm.h>
7  #include <unistd.h>          //LIBRERIAS VARIAS PARA EL FUNCIONAMIENTO DEL PROGRAMA
8
9  #define TAM_MEM 27
10
11 void reciboMatrizABS ();
12 void sumar ();
13 void enviarMatrizS ();
14
15 // RECIBO PARA LAS MATRICES AS Y BS
16
17 int shmidABS;
18 key_t llaveABS = 5682;
19 char *shmABS, *sABS;
20
21 // ENVIO PARA LA MATRIZ S
22
23 int shmidS;
24 key_t llaveS = 5684;
25 char *shmS, *sS;
26
27 // MATRICES
28
29 int matrizAS[10][10], matrizBS[10][10], matrizS[10][10];
30
31 // VARIABLES VARIAS
32
33 int aux1 = 0, aux2 = 0, aux3 = 0, aux4 = 0, i = 0, j = 0;
34
35 int main(int argc, char *argv[]){
36
37     reciboMatrizABS();
38     sumar();
39     enviarMatrizS();
40
41     exit(0);
42 }
43
44 void reciboMatrizABS (){
45 // CODIGO QUE ESTABLECE LA CONEXION PARA LAS MATRICES AS Y BS ***
46
47
48
49 //CODIGO QUE GUARDA LAS MATRICES AS Y BS ***
50 }
51
52 void sumar (){
53 //CODIGO QUE SUMA LAS MATRICES AS Y BS
54
55     for (i = 0; i < 10; i++) {
56
57         for (j = 0; j < 10; j++) {
58
59             matrizS[i][j] = matrizAS[i][j] + matrizBS[i][j];
60
61         }
62     }
63 }
64
65 void enviarMatrizS (){
66 // CODIGO QUE ESTABLECE CONEXION PARA LA MATRIZ S ***
67
68
69
70 // CODIGO PARA ENVIAR MATRIZ S ***
71 }
```

Ejecución

```
Jomiantc@Jomiantc-SVF15215CLW: ~/Escritorio
Jomiantc@Jomiantc-SVF15215CLW:~/Escritorio$ gcc Nieto.c -o NIETO.exe
Jomiantc@Jomiantc-SVF15215CLW:~/Escritorio$ gcc Hijo.c -o HIJO.exe
Jomiantc@Jomiantc-SVF15215CLW:~/Escritorio$ gcc Padre.c -o PADRE.exe
Jomiantc@Jomiantc-SVF15215CLW:~/Escritorio$ ./PADRE.exe

Matriz inversa de la multiplicacion
0.17, -0.67, 0.60, -0.19, -1.57, 0.78, -1.10, 0.41, 1.00, 0.75,
-0.01, 0.63, -0.99, 0.48, 1.63, -0.86, 0.90, -0.34, -0.95, -0.69,
0.23, -0.33, 0.22, 0.04, -0.77, 0.23, -0.76, 0.15, 0.62, 0.50,
-0.21, 0.52, -0.55, 0.06, 1.38, -0.55, 1.13, -0.30, -0.97, -0.70,
0.14, -1.09, 1.31, -0.45, -2.56, 1.32, -1.76, 0.51, 1.69, 1.25,
-0.03, 0.02, 0.19, -0.11, -0.13, 0.07, -0.01, 0.04, -0.00, -0.02,
-0.28, 1.10, -1.20, 0.31, 2.65, -1.26, 1.96, -0.53, -1.80, -1.31,
0.21, -0.89, 1.08, -0.35, -2.32, 1.10, -1.64, 0.52, 1.54, 1.07,
-0.14, 0.68, -0.89, 0.34, 1.53, -0.74, 1.03, -0.30, -1.00, -0.72,
0.01, -0.33, 0.60, -0.24, -0.70, 0.35, -0.36, 0.05, 0.44, 0.30,

Matriz inversa de la Suma
0.01, -0.07, 0.09, -0.09, -0.10, 0.07, -0.11, -0.08, 0.14, 0.17,
-0.01, 0.01, -0.17, -0.05, 0.19, -0.01, 0.03, 0.17, -0.02, -0.11,
0.10, -0.07, 0.09, -0.10, -0.16, 0.03, 0.10, 0.07, -0.02, -0.02,
-0.07, 0.05, 0.09, 0.03, -0.14, 0.06, -0.06, -0.10, 0.13, 0.04,
-0.10, 0.11, 0.19, -0.06, -0.43, 0.14, 0.22, -0.10, -0.25, 0.25,
-0.12, 0.02, -0.06, 0.12, 0.02, -0.03, 0.12, -0.03, -0.11, 0.06,
-0.01, 0.01, -0.10, 0.09, -0.07, 0.07, -0.11, 0.09, 0.04, 0.01,
0.14, 0.05, -0.10, 0.02, 0.29, -0.23, -0.10, 0.05, 0.02, -0.11,
0.16, -0.12, -0.20, 0.09, 0.65, -0.18, -0.25, -0.20, 0.32, -0.23,
-0.09, 0.02, 0.20, -0.03, -0.28, 0.11, 0.15, 0.15, -0.23, -0.00,
Jomiantc@Jomiantc-SVF15215CLW:~/Escritorio$
```

Archivo de texto creado

```
Abrir  *INVERSAS.txt  Guardar
~/Escritorio

1 Matriz inversa de la multiplicacion:
2 0.17, -0.67, 0.60, -0.19, -1.57, 0.78, -1.10, 0.41, 1.00, 0.75,
3 -0.01, 0.63, -0.99, 0.48, 1.63, -0.86, 0.90, -0.34, -0.95, -0.69,
4 0.23, -0.33, 0.22, 0.04, -0.77, 0.23, -0.76, 0.15, 0.62, 0.50,
5 -0.21, 0.52, -0.55, 0.06, 1.38, -0.55, 1.13, -0.30, -0.97, -0.70,
6 0.14, -1.09, 1.31, -0.45, -2.56, 1.32, -1.76, 0.51, 1.69, 1.25,
7 -0.03, 0.02, 0.19, -0.11, -0.13, 0.07, -0.01, 0.04, -0.00, -0.02,
8 -0.28, 1.10, -1.20, 0.31, 2.65, -1.26, 1.96, -0.53, -1.80, -1.31,
9 0.21, -0.89, 1.08, -0.35, -2.32, 1.10, -1.64, 0.52, 1.54, 1.07,
10 -0.14, 0.68, -0.89, 0.34, 1.53, -0.74, 1.03, -0.30, -1.00, -0.72,
11 0.01, -0.33, 0.60, -0.24, -0.70, 0.35, -0.36, 0.05, 0.44, 0.30,
12
13 Matriz inversa de la suma:
14 0.01, -0.07, 0.09, -0.09, -0.10, 0.07, -0.11, -0.08, 0.14, 0.17,
15 -0.01, 0.01, -0.17, -0.05, 0.19, -0.01, 0.03, 0.17, -0.02, -0.11,
16 0.10, -0.07, 0.09, -0.10, -0.16, 0.03, 0.10, 0.07, -0.02, -0.02,
17 -0.07, 0.05, 0.09, 0.03, -0.14, 0.06, -0.06, -0.10, 0.13, 0.04,
18 -0.10, 0.11, 0.19, -0.06, -0.43, 0.14, 0.22, -0.10, -0.25, 0.25,
19 -0.12, 0.02, -0.06, 0.12, 0.02, -0.03, 0.12, -0.03, -0.11, 0.06,
20 -0.01, 0.01, -0.10, 0.09, -0.07, 0.07, -0.11, 0.09, 0.04, 0.01,
21 0.14, 0.05, -0.10, 0.02, 0.29, -0.23, -0.10, 0.05, 0.02, -0.11,
22 0.16, -0.12, -0.20, 0.09, 0.65, -0.18, -0.25, -0.20, 0.32, -0.23,
23 -0.09, 0.02, 0.20, -0.03, -0.28, 0.11, 0.15, 0.15, -0.23, -0.00,

Texto plano  Anchura del tabulador: 8  Ln 24, Col 1  INS
```

Windows

Programa Padre.c

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <windows.h>
4
5  #define TAM_MEM 600
6
7  void envioMatrizAB ();
8  void reciboMatrizS ();
9  void reciboMatrizM ();
10 void inversa();
11
12 // ENVIO PARA LAS MATRICES A Y B
13
14 HANDLE hArchMapeoAB;
15 char *idMemCompartidaAB = "MATRIZAB";
16 char *apDatosAB, *apTrabajoAB;
17
18 // RECIBO PARA LA MATRIZ S
19
20 HANDLE hArchMapeoS;
21 char *idMemCompartidaS = "MATRIZS";
22 char *apDatosS, *apTrabajoS;
23
24 // RECIBO PARA LA MATRIZ M
25
26 HANDLE hArchMapeoM;
27 char *idMemCompartidaM = "MATRIZM";
28 char *apDatosM, *apTrabajoM;
29
30 // MATRICES
31
32 int matrizA[10][10] = { ...
33 };
34
35 int matrizB[10][10] = { ...
36 };
37
38 double identidadS[10][10] = { ...
39 };
40
41 double identidadM[10][10] = { ...
42 };
43
44 int matrizM[10][10], matrizS[10][10];
45
46 // VARIABLES VARIAS
47
48 int i = 0, j = 0, k = 0, aux1 = 0, aux2 = 0, aux3 = 0, aux4 = 0;
49
50 double auxS, pivotes, saveS[10][10], auxM, pivoteM, saveM[10][10];
```

```

96  int main(int argc, char *argv[]){
97  // CODIGO PARA LA CREACION DEL PROCESO HIJO
98
99      STARTUPINFO si;
100     PROCESS_INFORMATION pi;
101     ZeroMemory(&si, sizeof(si));
102     si.cb = sizeof(si);
103     ZeroMemory(&pi, sizeof(pi));
104
105     if(!CreateProcess(NULL, "HIJO1", NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi)){
106
107         printf( "Fallo al invocar CreateProcess (%d)\n", GetLastError() );
108         return;
109     }
110
111     envioMatrizAB();
112
113     reciboMatrizM();
114     reciboMatrizS();
115
116     inversa();
117
118     WaitForSingleObject(pi.hProcess, INFINITE);
119
120     CloseHandle(pi.hProcess);
121     CloseHandle(pi.hThread);
122 }
123
124 void envioMatrizAB (){
125 // CODIGO QUE ESTABLECE CONEXION PARA LAS MATRICES A Y B
126
127     if((hArchMapeoAB=CreateFileMapping( INVALID_HANDLE_VALUE, NULL, PAGE_READWRITE, 0,
128                                         TAM_MEM, idMemCompartidaAB)) == NULL){
129
130         printf("No se mapeo la memoria compartida: (%i)\n", GetLastError());
131         exit(-1);
132     }
133
134     if((apDatosAB=(char *)MapViewOfFile(hArchMapeoAB, FILE_MAP_ALL_ACCESS, 0, 0,
135                                         TAM_MEM)) == NULL){
136
137         printf("No se creo la memoria compartida: (%i)\n", GetLastError());
138         CloseHandle(hArchMapeoAB);
139         exit(-1);
140     }
141
142 // CODIGO PARA ENVIAR MATRICES A Y B
143
144     apTrabajoAB = apDatosAB;
145
146     for (i = 0; i < 10; i++){
147         for (j = 0; j < 10; j++){
148             *apTrabajoAB++ = matrizA[i][j];
149         }
150
151         for (j = 0; j < 10; j++){
152             *apTrabajoAB++ = matrizB[i][j];
153         }
154
155         *apTrabajoAB = '\0';
156
157         while (*apDatosAB != '*'){
158             sleep(1);
159         }
160
161         UnmapViewOfFile(apDatosAB);
162         CloseHandle(hArchMapeoAB);
163     }
164 }

```



```

166 void reciboMatrizS (){
167 // CODIGO QUE ESTABLECE LA CONEXION PARA LAS MATRICES M
168
169     if((hArchMapeoS=OpenFileMapping(FILE_MAP_ALL_ACCESS, FALSE, idMemCompartidaS)) == NULL){
170
171         printf("No se padre archivo de mapeo de la memoria compartidas: (%i)\n", GetLastError());
172         exit(-1);
173     }
174
175     if((apDatosS=(char *)MapViewOfFile(hArchMapeoS, FILE_MAP_ALL_ACCESS,0, 0, TAM_MEM)) == NULL){
176
177         printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
178         CloseHandle(hArchMapeoS);
179         exit(-1);
180     }
181
182 // CODIGO QUE GUARDA LA MATRIZ M
183
184     for (apTrabajos = apDatosS; *apTrabajos != '\0'; apTrabajos++){
185
186         matrizS[aux3][aux4] = *apTrabajos;
187         aux4++;
188
189         if (aux4 == 10){
190
191             aux4 = 0;
192             aux3++;
193         }
194     }printf("\nMatrices A y B sumadas: \n");
195
196     for (i = 0; i < 10; i++){
197
198         for (j = 0; j < 10; j++){
199
200             printf("%d,", matrizS[i][j]);
201         }
202         printf("\n");
203     }
204
205     *apDatosS = '*';
206     UnmapViewOfFile(apDatosS);
207     CloseHandle(hArchMapeoS);
208 }
209
210 void reciboMatrizM (){
211 // CODIGO QUE ESTABLECE LA CONEXION PARA LAS MATRICES M
225
226 // CODIGO QUE GUARDA LA MATRIZ M
252 }

```



```

254 void inversa (){
255 // CODIGO QUE CALCULA LA INVERSA DE LAS MATRICES M Y S
256 for (i = 0; i < 10; i++){
257
258     for (j = 0; j < 10; j++){
259
260         saveM[i][j] = matrizM[i][j];
261         saveS[i][j] = matrizS[i][j];
262     }
263 }
264
265 for (i = 0; i < 10; i++){
266
267     pivoteM = saveM[i][i];
268     pivoteS = saveS[i][i];
269
270     for (k = 0; k < 10; k++){
271
272         saveM[i][k] = saveM[i][k]/pivoteM;
273         identidadM[i][k] = identidadM[i][k]/pivoteM;
274         saveS[i][k] = saveS[i][k]/pivoteS;
275         identidadS[i][k] = identidadS[i][k]/pivoteS;
276
277     }
278
279     for (j = 0; j < 10; j++){
280
281         if (i != j) {
282
283             auxM = saveM[j][i];
284             auxS = saveS[j][i];
285
286             for (k = 0; k < 10; k++){
287
288                 saveM[j][k] = saveM[j][k] - auxM*saveM[i][k];
289                 identidadM[j][k] = identidadM[j][k] - auxM*identidadM[i][k];
290                 saveS[j][k] = saveS[j][k] - auxS*saveS[i][k];
291                 identidadS[j][k] = identidadS[j][k] - auxS*identidadS[i][k];
292             }
293         }
294     }
295 }
296
297 FILE *fichero;
298 fichero = fopen("INVERSAS.txt","w");
299 fprintf(fichero, "Matriz inversa de la multiplicacion: \n");
300
301 for (i = 0; i < 10; i++){
302
303     for (j = 0; j < 10; j++){
304
305         fprintf(fichero, "%0.2lf, ", identidadM[i][j]);
306
307     }
308     fprintf(fichero, "\n");
309 }
310
311 fprintf(fichero, "\nMatriz inversa de la suma: \n");
312
313 for (i = 0; i < 10; i++){
314
315     for (j = 0; j < 10; j++){
316
317         fprintf(fichero, "%0.2lf, ", identidadS[i][j]);
318
319     }
320     fprintf(fichero, "\n");
321 }
322
323 fclose(fichero);
324 exit(0);
325 }

```

Código Hijo.c

```
1  #include <stdio.h>
2  #include <windows.h>
3
4  #define TAM_MEM 600
5
6  void saveMatrizAB ();
7  void sendMatrizABS ();
8  void multiplicacionMatrizAB ();
9  void sendMatrizM ();
10
11 // RECIBO PARA LAS MATRICES A Y B
12
13 HANDLE hArchMapeoAB;
14 char *idMemCompartidaAB = "MATRIZAB";
15 char *apDatosAB, *apTrabajoAB;
16
17 // ENVIO PARA LAS MATRICES AS Y BS
18
19 HANDLE hArchMapeoABS;
20 char *idMemCompartidaABS = "MATRIZABS";
21 char *apDatosABS, *apTrabajoABS;
22
23 // ENVIO PARA LA MATRIZ M
24
25 HANDLE hArchMapeoM;
26 char *idMemCompartidaM = "MATRIZM";
27 char *apDatosM, *apTrabajoM;
28
29 // MATRICES
30
31 int matrizA[10][10], matrizB[10][10], matrizM[10][10];
32
33 // VARIABLES VARIAS
34
35 int aux1 = 0, aux2 = 0, aux3 = 0, aux4 = 0, suma = 0, i, j, a;
36
37 int main(int argc, char *argv[]){
38
39     STARTUPINFO si;
40     PROCESS_INFORMATION pi;
41     ZeroMemory(&si, sizeof(si));
42     si.cb = sizeof(si);
43     ZeroMemory(&pi, sizeof(pi));
44
45     if(!CreateProcess(NULL, "NIETO1", NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi)){
46
47         printf( "Fallo al invocar CreateProcess (%d)\n", GetLastError() );
48         return;
49     }
50
51     saveMatrizAB();
52     multiplicacionMatrizAB();
53     sendMatrizM();
54     sendMatrizABS();
55
56     WaitForSingleObject(pi.hProcess, INFINITE);
57
58     CloseHandle(pi.hProcess);
59     CloseHandle(pi.hThread);
60     exit(0);
61 }
```

```

63 void saveMatrizAB (){
64 // CODIGO QUE ESTABLECE LA CONEXION PARA LAS MATRICES A Y B ***
78
79 //CODIGO QUE GUARDA LAS MATRICES A Y B ***
111 }
112
113 void sendMatrizABS (){
114 // CODIGO QUE ESTABLECE CONEXION PARA LAS MATRICES AS Y BS ***
130
131 // CODIGO PARA ENVIAR MATRICES AS Y BS ***
154 }
155
156 void multiplicacionMatrizAB (){
157 //CODIGO QUE MULTIPLICA LAS MATRICES A Y B
158     for (a = 0; a < 10; a++) {
159
160         for (i = 0; i < 10; i++) {
161
162             suma = 0;
163
164             for (j = 0; j < 10; j++) {
165
166                 suma += matrizA[i][j] * matrizB[j][a];
167             }
168
169             matrizM[i][a] = suma+a;
170         }
171     }
172 }
173
174 void sendMatrizM (){
175 // CODIGO PARA ENVIAR DE VUELTA A PADRE ***
191
192 // CODIGO PARA ENVIAR MATRICES M ***
210 }

```

Código Nieto.c

```
1  #include <stdio.h>
2  #include <windows.h>
3
4  #define TAM_MEM 600
5
6  void saveMatrizABS ();
7  void sumaMatrizABS ();
8  void sendMatrizS ();
9
10 // RECIBO PARA LAS MATRICES A Y B
11
12 HANDLE hArchMapeoABS;
13 char *idMemCompartidaABS = "MATRIZABS";
14 char *apDatosABS, *apTrabajoABS;
15
16 // ENVIO PARA LA MATRIZ S
17
18 HANDLE hArchMapeoS;
19 char *idMemCompartidas = "MATRIZS";
20 char *apDatosS, *apTrabajos;
21
22 // MATRICES
23
24 int matrizA[10][10], matrizB[10][10], matrizS[10][10];
25
26 // VARIABLES VARIAS
27
28 int aux1 = 0, aux2 = 0, aux3 = 0, aux4 = 0, suma = 0, i, j, a;
29
30 int main(void){
31
32     saveMatrizABS();
33     sumaMatrizABS();
34     sendMatrizS();
35
36     exit(0);
37 }
38
39 void saveMatrizABS (){
40 // CODIGO QUE ESTABLECE LA CONEXION PARA LAS MATRICES AS Y BS ***
54
55 //CODIGO QUE GUARDA LAS MATRICES A Y B ***
87 }
88
89 void sumaMatrizABS (){
90 //CODIGO QUE SUMA LAS MATRICES AS Y BS ***
99 }
100
101 void sendMatrizS (){
102 // CODIGO QUE ESTABLECE CONEXION PARA LAS MATRICES AS Y BS ***
118
119 // CODIGO PARA ENVIAR MATRICES A Y B ***
138 }
```

Administrador: C:\Windows\system32\cmd.exe

C:\Users\JomianTC\Desktop>PADRE1.exe

Matriz A y B multiplicada

63,76,88,93,61,94,65,68,86,89,
49,63,79,74,61,82,65,63,82,74,
49,66,64,56,62,73,69,53,65,71,
57,69,66,55,65,75,72,54,78,81,
49,76,72,71,71,77,76,64,65,70,
53,71,75,81,62,85,59,60,77,74,
63,69,84,75,66,78,80,69,91,92,
64,83,79,67,81,84,99,73,77,84,
44,64,66,65,57,58,71,70,68,72,
52,59,80,65,68,76,72,51,78,72,

Matrices A y B sumadas:

3,4,8,3,8,3,7,5,7,4,
2,8,3,9,8,2,4,7,5,4,
6,4,4,6,4,5,2,6,6,9,
3,2,6,5,4,8,7,5,6,6,
6,7,3,4,6,5,3,5,7,7,
4,8,4,4,9,2,7,2,9,7,
3,7,9,7,5,8,2,4,5,3,
6,6,5,5,2,6,4,6,3,6,
4,6,7,9,2,6,2,4,3,2,
8,5,4,3,5,7,4,6,4,4,

Matriz inversa de la multiplicacion

0.40, -0.78, 0.59, -0.08, -1.98, 0.77, -1.64, 0.55, 1.37, 1.03,
-0.02, 0.64, -0.99, 0.47, 1.65, -0.86, 0.93, -0.34, -0.97, -0.71,
0.54, -0.48, 0.21, 0.19, -1.33, 0.21, -1.50, 0.33, 1.12, 0.89,
-0.51, 0.66, -0.54, -0.09, 1.90, -0.54, 1.82, -0.47, -1.44, -1.07,
0.34, -1.19, 1.30, -0.35, -2.91, 1.31, -2.23, 0.63, 2.01, 1.50,
-0.08, 0.05, 0.19, -0.13, -0.05, 0.07, 0.09, 0.01, -0.07, -0.07,
-0.66, 1.29, -1.18, 0.12, 3.33, -1.25, 2.87, -0.75, -2.42, -1.79,
0.49, -1.03, 1.07, -0.21, -2.82, 1.09, -2.31, 0.68, 1.99, 1.42,
-0.33, 0.77, -0.88, 0.25, 1.87, -0.73, 1.47, -0.41, -1.30, -0.96,
0.01, -0.34, 0.60, -0.23, -0.71, 0.35, -0.38, 0.06, 0.46, 0.31,

Matriz inversa de la Suma

0.01, -0.07, 0.08, -0.09, -0.09, 0.07, -0.11, -0.08, 0.14, 0.17,
-0.01, 0.01, -0.17, -0.05, 0.18, -0.01, 0.03, 0.17, -0.02, -0.11,
0.11, -0.06, 0.08, -0.09, -0.13, 0.01, 0.09, 0.07, -0.02, -0.04,
-0.08, 0.05, 0.10, 0.03, -0.16, 0.07, -0.05, -0.11, 0.13, 0.05,
-0.12, 0.11, 0.21, -0.06, -0.47, 0.17, 0.23, -0.10, -0.25, 0.27,
-0.14, 0.01, -0.04, 0.11, -0.02, 0.01, 0.14, -0.04, -0.11, 0.08,
-0.01, 0.01, -0.10, 0.09, -0.07, 0.08, -0.11, 0.09, 0.04, 0.01,
0.17, 0.06, -0.12, 0.02, 0.34, -0.26, -0.11, 0.06, 0.02, -0.13,
0.19, -0.11, -0.22, 0.10, 0.71, -0.23, -0.27, -0.19, 0.32, -0.25,
-0.11, 0.02, 0.21, -0.03, -0.32, 0.14, 0.16, 0.14, -0.23, 0.01,

C:\Users\JomianTC\Desktop>

Matriz inversa de la multiplicacion:

0.40, -0.78, 0.59, -0.08, -1.98, 0.77, -1.64, 0.55, 1.37, 1.03,
-0.02, 0.64, -0.99, 0.47, 1.65, -0.86, 0.93, -0.34, -0.97, -0.71,
0.54, -0.48, 0.21, 0.19, -1.33, 0.21, -1.50, 0.33, 1.12, 0.89,
-0.51, 0.66, -0.54, -0.09, 1.90, -0.54, 1.82, -0.47, -1.44, -1.07,
0.34, -1.19, 1.30, -0.35, -2.91, 1.31, -2.23, 0.63, 2.01, 1.50,
-0.08, 0.05, 0.19, -0.13, -0.05, 0.07, 0.09, 0.01, -0.07, -0.07,
-0.66, 1.29, -1.18, 0.12, 3.33, -1.25, 2.87, -0.75, -2.42, -1.79,
0.49, -1.03, 1.07, -0.21, -2.82, 1.09, -2.31, 0.68, 1.99, 1.42,
-0.33, 0.77, -0.88, 0.25, 1.87, -0.73, 1.47, -0.41, -1.30, -0.96,
0.01, -0.34, 0.60, -0.23, -0.71, 0.35, -0.38, 0.06, 0.46, 0.31,

Matriz inversa de la suma:

0.01, -0.07, 0.08, -0.09, -0.09, 0.07, -0.11, -0.08, 0.14, 0.17,
-0.01, 0.01, -0.17, -0.05, 0.18, -0.01, 0.03, 0.17, -0.02, -0.11,
0.11, -0.06, 0.08, -0.09, -0.13, 0.01, 0.09, 0.07, -0.02, -0.04,
-0.08, 0.05, 0.10, 0.03, -0.16, 0.07, -0.05, -0.11, 0.13, 0.05,
-0.12, 0.11, 0.21, -0.06, -0.47, 0.17, 0.23, -0.10, -0.25, 0.27,
-0.14, 0.01, -0.04, 0.11, -0.02, 0.01, 0.14, -0.04, -0.11, 0.08,
-0.01, 0.01, -0.10, 0.09, -0.07, 0.08, -0.11, 0.09, 0.04, 0.01,
0.17, 0.06, -0.12, 0.02, 0.34, -0.26, -0.11, 0.06, 0.02, -0.13,
0.19, -0.11, -0.22, 0.10, 0.71, -0.23, -0.27, -0.19, 0.32, -0.25,
-0.11, 0.02, 0.21, -0.03, -0.32, 0.14, 0.16, 0.14, -0.23, 0.01,