



Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 6 - Comunicación inter procesos (IPC) en Linux y Windows

Unidad de aprendizaje: Sistemas Operativos

Grupo: 2CM8

Alumnos(a):

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

Profesor(a):

Cortes Galicia Jorge

26 de noviembre 2018

Índice

1 Competencias	2
2 Desarrollo	2
2.1 Puntos a observar y reportar	2
2.1.1 Sección Linux:	2
2.1.2 Sección Windows:	5
2.2 Explicación general de los programas	7
2.3 Códigos fuente de los programas desarrollados	8
2.3.1 Sección Linux	8
2.3.2 Sección Windows	14
2.4 Pantallas de ejecución de los programas desarrollados	26
2.4.1 Sección Linux:	26
2.4.2 Sección Windows:	30
3 Observaciones	35
4 Análisis Crítico	37
5 Conclusiones	37

1. Competencias

El alumno comprende el funcionamiento de las tuberías (pipes) sin nombre y de la memoria compartida como mecanismos de comunicación entre procesos tanto en el sistema operativo Linux como Windows para el desarrollo de aplicaciones concurrentes con soporte de comunicación.

- Revisión de tuberías y memoria compartida tanto en Linux como en Windows.
- Revisión de las llamadas al sistema para tuberías y memoria compartida en ambos sistemas operativos.
- Desarrollo de aplicaciones concurrentes que requieren comunicación en Linux y Windows.

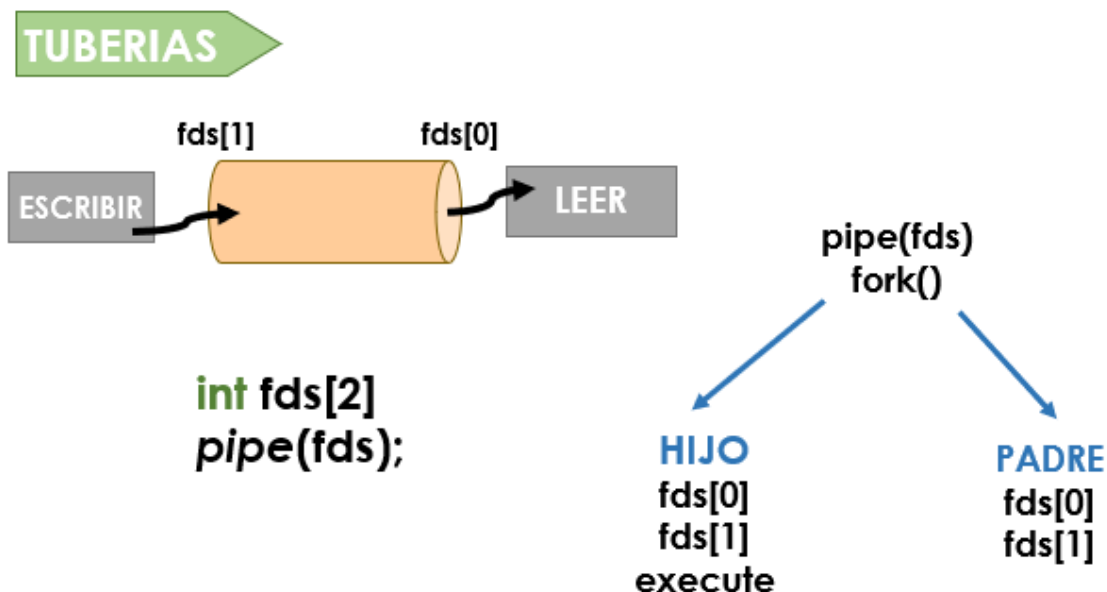
2. Desarrollo

2.1. Puntos a observar y reportar

2.1.1. Sección Linux:

Investigación de los siguientes comandos:

✓ **pipe()**



Permite enviar la salida de uno comando a otro. Una tubería, como suele decirse, puede redirigir la salida estándar, la entrada o el error de un proceso a otro para su posterior procesamiento.

La sintaxis para el comando es con el carácter | entre dos comandos:

Comando 1 | Comando 2 | ... | Comando n

No se puede acceder a la tubería a través de otra sesión, se crea temporalmente para acomodar la ejecución del **Comando 1** y redirigir la salida estándar. Se elimina después de la ejecución exitosa.

Una tubería con nombre puede durar hasta que el sistema esté en funcionamiento o hasta que se elimine. Es un archivo especial que sigue el mecanismo **FIFO** (Primero en entrar, primero en salir). Se puede utilizar como un archivo normal, es decir, puede escribir, leer, abrir o cerrar. Para crear una tubería con nombre se usa el comando:

mkfifo <nombre_tubería>

✓ shmget()

Asigna un segmento de memoria compartida de System V. Retorna el identificador del segmento de memoria compartida de System V asociado con el valor del argumento **key**. Puede ser usado ya sea para obtener el identificador de un segmento de memoria compartida creada previamente (cuando shmglh es cero y la clave no tiene valor IPCPRIVATE), o para crear un nuevo conjunto.

Un nuevo segmento de memoria compartida, con tamaño igual que el valor de **size** redondeado a un múltiplo de **PAGESIZE**, se crea si la clave tiene el valor **IPCPRIVATE**.

✓ shmat() Adjunta el segmento de memoria compartida asociado con el identificador de memoria compartida especificado por shmid al espacio de direcciones del proceso de llamado.

✓ Punto 2: Tuberías en Linux

```

1  #include <stdio.h>
2  #include <unistd.h>
3  #include <string.h>
4  #include <stdlib.h>
5  #define VALOR 1
6
7  int main(void)
8  {
9      int desc_arch[2];
10     char bufer[100];
11     if(pipe(desc_arch) != 0)
12         exit(1);
13     if(fork()==0)
14     {
15         while(VALOR)
16         {
17             read(desc_arch[0], bufer, sizeof(bufer));
18             printf("Se recibio: %s\n", bufer);
19         }
20     }
21     while(VALOR)
22     {
23         gets(bufer);
24         write(desc_arch[1], bufer, strlen(bufer)+1);
25     }
26 }
```

✓ Punto 5: Memoria compartida en Linux

CLIENTE:

```

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

```

SERVIDOR:

```

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

```

```

30     while(*shm != '*')
31         sleep(1);
32     exit(0);
33 }

```

2.1.2. Sección Windows:

✓ Punto 3: Tuberías en Windows

```

1  //Compilar: gcc 3.c -o 3
2  //Ejecutar: 3 hijo
3  #include <stdio.h>
4  #include <windows.h>
5  #include <string.h>
6
7  int main(int argc, char *argv[])
8  {
9      char mensaje[] = "Tuberias en Windows";
10     DWORD escritos;
11     HANDLE hLecturaPipe, hEscrituraPipe;
12     PROCESS_INFORMATION piHijo;
13     STARTUPINFO siHijo;
14     SECURITY_ATTRIBUTES pipeSeg = {sizeof(SEcurity_ATTRIBUTES), NULL, TRUE};
15
16     /*Obtencion de la informacion para la inicializacion del proceso hijo*/
17     GetStartupInfo(&siHijo);
18     /*Creación de la tubería sin nombre*/
19     CreatePipe(&hLecturaPipe, &hEscrituraPipe, &pipeSeg, 0);
20     /*Escritura de la tubería sin nombre*/
21     WriteFile(hEscrituraPipe, mensaje, strlen(mensaje)+1, &escritos, NULL);
22
23     siHijo.hStdInput = hLecturaPipe;
24     siHijo.hStdError = GetStdHandle(STD_ERROR_HANDLE);
25     siHijo.hStdOutput = GetStdHandle(STD_OUTPUT_HANDLE);
26     siHijo.dwFlags = STARTF_USESTDHANDLES;
27     CreateProcess(NULL, argv[1], NULL, NULL,
28         TRUE, /*Hereda el proceso hijo los manejadores de la tubería del padre*/
29         0, NULL, NULL, &siHijo, &piHijo);
30
31     WaitForSingleObject(piHijo.hProcess, INFINITE);
32     printf("Mensaje recibido en el proceso hijo, termina el proceso padre\n");
33     CloseHandle(hLecturaPipe);
34     CloseHandle(hEscrituraPipe);
35     CloseHandle(piHijo.hThread);
36     CloseHandle(piHijo.hProcess);
37     return 0;
38 }

```

PROGRAMA HIJO

```

1  /*Programa 3_hijo.c*/
2  //Compilar: gcc 3_hijo.c -o hijo
3  #include <stdio.h>
4  #include <windows.h>
5
6  int main()
7  {
8      char mensaje[20];
9      DWORD leidos;
10     HANDLE hStdIn = GetStdHandle(STD_INPUT_HANDLE);
11     SECURITY_ATTRIBUTES pipeSeg = {sizeof(SEcurity_ATTRIBUTES), NULL, TRUE};
12
13     /*Lectura desde la tubería sin nombre*/

```

```

14     ReadFile(hStdIn, mensaje, sizeof(mensaje), &leidos, NULL);
15     printf("Mensaje recibido del proceso padre: %s\n", mensaje);
16     CloseHandle(hStdIn);
17     printf("Termina el proceso hijo, continua el proceso padre\n");
18     return 0;
19 }

```

✓ Punto 6: Memoria compartida en Windows

CLIENTE

```

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 = "Memoria Compartida";
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 el mapo 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
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
29     for(apTrabajo = apDatos; *apTrabajo != '\0'; apTrabajo++)
30         putchar(*apTrabajo);
31     putchar('\n');
32     *apDatos = '*';
33     UnmapViewOfFile(apDatos);
34     CloseHandle(hArchMapeo);
35     exit(0);
36 }

```

SERVIDOR

```

1  #include <windows.h>    /*Servidor de la memoria compartida*/
2  #include <stdio.h>      /*(ejecutar el servidor antes de ejecutar el client)*/
3  #define TAM_MEM 27     /*Tamaño de la memoria compartida en bytes*/
4  int main(void)
5  {
6      HANDLE hArchMapeo;
7      char *idMemCompartida = "Memoria Compartida";
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
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

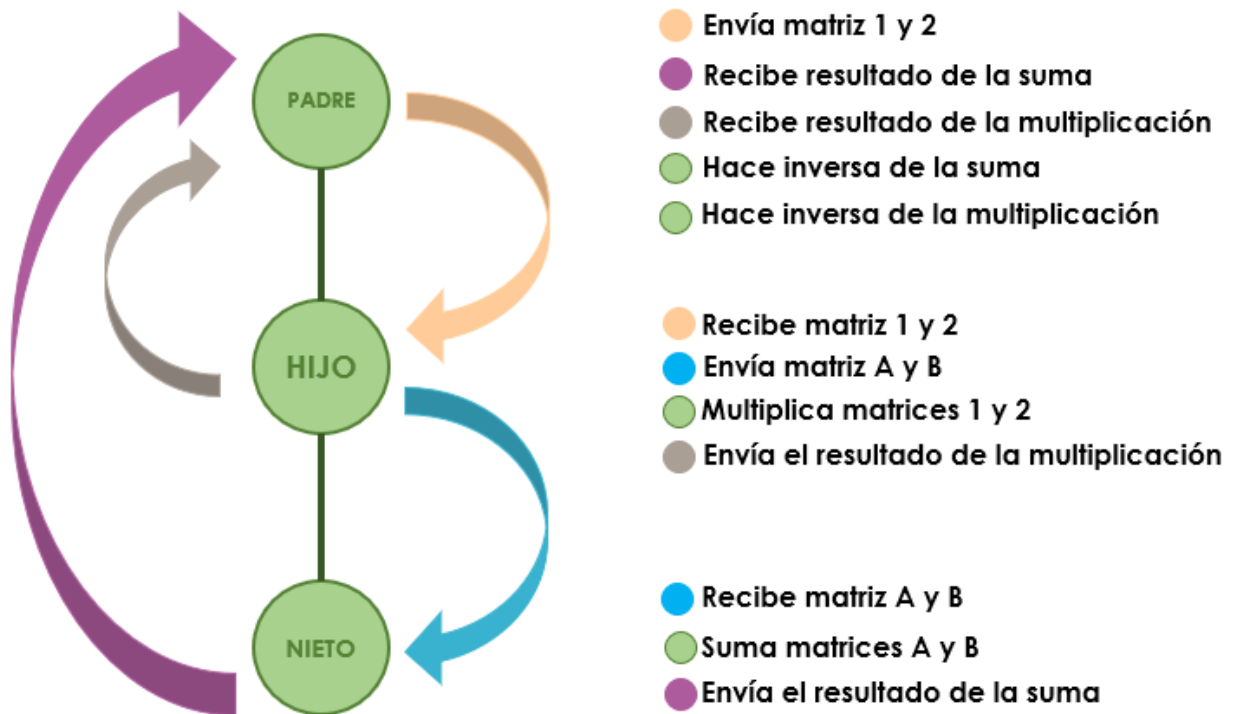
```

```

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
33 for(c = 'a'; c <= 'z'; c++)
34     *apTrabajo++ = c;
35 *apTrabajo = '\0';
36 while(*apDatos != '*')
37     Sleep(1);
38 UnmapViewOfFile(apDatos);
39 CloseHandle(hArchMapeo);
40 exit(0);
41 }

```

2.2. Explicación general de los programas



2.3. Códigos fuente de los programas desarrollados

2.3.1. Sección Linux

✓ Punto 4: Inversa de la suma y multiplicación de matrices con tuberías

```

1  #include <stdio.h>
2  #include <unistd.h>
3  #include <string.h>
4  #include <stdlib.h>
5  #include <fcntl.h>
6  #include "funciones.h"
7  char* leerDirectorio();
8
9  int main(void)
10 {
11     // CREAMOS DIRECTORIO
12     // Obtenemos el directorio desde la entrada de teclado
13     char* path = leerDirectorio();
14     //Llamada al sistema mkdir recibe la ruta del directorio a crear, y los permisos de escritura, lectura y
15     //↪ ejecución para cada tipo de usuario
16     //Retorna -1 si ocurrieron errores
17     if(mkdir(path, S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH) == -1)
18     {
19         perror(path);
20         exit(EXIT_FAILURE);
21     }
22
23     int n = 10, i, j;
24     int pipefd_0_1[2];
25     int pipefd_1_0[2];
26     int pipefd_1_2[2];
27     int pipefd_2_0[2];
28     double **matrixA, **matrixB, **mul, **suma, **inv1, **inv2;
29
30     // Inicializa las matrices.
31     matrixA = (double**)calloc(n,sizeof(double*));
32     for (i = 0; i < n; i++)
33         matrixA[i] = (double*)calloc(n,sizeof(double));
34
35     matrixB = (double**)calloc(n,sizeof(double*));
36     for (i = 0; i < n; i++)
37         matrixB[i] = (double*)calloc(n,sizeof(double));
38
39     mul = (double**)calloc(n,sizeof(double*));
40     for (i = 0; i < n; i++)
41         mul[i] = (double*)calloc(n,sizeof(double));
42
43     suma = (double**)calloc(n,sizeof(double*));
44     for (i = 0; i < n; i++)
45         suma[i] = (double*)calloc(n,sizeof(double));
46
47     inv1 = (double**)calloc(n,sizeof(double*));
48     for (i = 0; i < n; i++)
49         inv1[i] = (double*)calloc(n,sizeof(double));
50
51     inv2 = (double**)calloc(n,sizeof(double*));
52     for (i = 0; i < n; i++)
53         inv2[i] = (double*)calloc(n,sizeof(double));
54
55     if (pipe (pipefd_0_1) != 0 || pipe (pipefd_1_0) != 0 || pipe (pipefd_1_2) != 0 || pipe (pipefd_2_0) != 0)
56     {
57         exit(1);
58     }

```

```

58
59     if (fork())
60     { //0
61         srand(getpid());
62         llenar(matrixA, n);
63         llenar(matrixB, n);
64
65         printf("MATRIZ A\n"); imprimir(matrixA, n);
66         printf("\nMATRIZ B\n"); imprimir(matrixB, n);
67
68         for(i = 0; i < n; i++){
69             for(j = 0; j < n; j++){
70                 write(pipefd_0_1[1], &matrixA[i][j], sizeof(double));
71                 write(pipefd_0_1[1], &matrixB[i][j], sizeof(double));
72             }
73         }
74
75         for(i = 0; i < n; i++){
76             for(j = 0; j < n; j++){
77                 read(pipefd_1_0[0], &mul[i][j], sizeof(double));
78                 read(pipefd_2_0[0], &suma[i][j], sizeof(double));
79             }
80         }
81
82         printf("\nMultiplicacion realizada por el Proceso PADRE:\n");
83         imprimir(mul, n);
84         printf("Escribiendo archivo TXT de inversa multiplicacion...");
85         inversa(mul, inv1, n);
86         crearArchivo(inv1, n, "/inv_mul.txt", path);
87         printf("Listo\n\n");
88
89         printf("Suma realizada por el Proceso HIJO:\n");
90         imprimir(suma, n);
91         printf("Escribiendo archivo TXT de inversa suma...");
92         inversa(suma, inv2, n);
93         crearArchivo(inv2, n, "/inv_suma.txt", path);
94         printf("Listo\n\n");
95
96         exit(0);
97     }
98     else
99     {
100         if (fork())
101         { // 1
102             srand(getpid());
103             llenar(matrixA, n);
104             llenar(matrixB, n);
105             printf("\nMATRIZ C\n"); imprimir(matrixA, n);
106             printf("\nMATRIZ D\n"); imprimir(matrixB, n);
107
108             for(i = 0; i < n; i++){
109                 for(j = 0; j < n; j++){
110                     write(pipefd_1_2[1], &matrixA[i][j], sizeof(double));
111                     write(pipefd_1_2[1], &matrixB[i][j], sizeof(double));
112
113                     read(pipefd_0_1[0], &matrixA[i][j], sizeof(double));
114                     read(pipefd_0_1[0], &matrixB[i][j], sizeof(double));
115                 }
116             }
117
118             multiplicar(matrixA, matrixB, mul, n);
119
120             for(i = 0; i < n; i++){
121                 for(j = 0; j < n; j++){
122                     write(pipefd_1_0[1], &mul[i][j], sizeof(double));
123                 }

```

```

124     }
125     exit(0);
126 }
127 else
128 { // 2
129     for(i = 0; i < n; i++){
130         for(j = 0; j < n; j++){
131             read(pipefd_1_2[0], &matrixA[i][j], sizeof(double));
132             read(pipefd_1_2[0], &matrixB[i][j], sizeof(double));
133         }
134     }
135
136     sumar(matrixA, matrixB, suma, n);
137     for(i = 0; i < n; i++){
138         for(j = 0; j < n; j++){
139             write(pipefd_2_0[1], &suma[i][j], sizeof(double));
140         }
141     }
142     }
143     exit(0);
144 }
145 }
146 }
147
148 char* leerDirectorio()
149 {
150     char* directorio = (char*)calloc(2000,sizeof(char));
151     printf("Ingresa el nuevo directorio: ");
152     scanf("%s", directorio);
153     return directorio;
154 }

```

✓ Punto 7: Inversa de la suma y multiplicación de matrices con memoria compartida

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include <stdbool.h>
5  #include <math.h>
6  #include <sys/wait.h>
7  #include <sys/types.h>
8  #include <sys/stat.h>
9  #include <fcntl.h>
10 #include <errno.h>
11 #include <unistd.h>
12 #include <string.h>
13 #include <sys/ipc.h>
14 #include <sys/shm.h>
15 #include "funciones.h"
16 #define TAM_MEM 27
17
18 void escribir(double **matriz, int n, int clave);
19 void leer(double **matriz, int n, int clave);
20 char* leerDirectorio();
21 int *shm, *z, c, d;
22
23 int main()
24 {
25     // CREAM DIRECTORIO
26     // Obtenemos el directorio desde la entrada de teclado
27     char* path = leerDirectorio();
28     //Llamda al sistema mkdir recibe la ruta del directorio a crear, y los permisos de escritura, lectura y
29     ↪ ejecución para cada tipo de usuario
30     //Retorna -1 si ocurrieron errores
31     if(mkdir(path, S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH) ==-1)

```

```

31     {
32         perror(path);
33         exit(EXIT_FAILURE);
34     }
35
36     int i, n, shmid, pid;
37     double **matriz1, **matriz2, **matriz3, **matriz4, **suma, **mul, **inv1, **inv2;
38     n = 10; // Tam de la matriz cuadrada
39     key_t llave = 5678;
40
41     // Inicializa las matrices.
42     matriz1 = (double**)calloc(n,sizeof(double*));
43     for (i = 0; i < n; i++)
44         matriz1[i] = (double*)calloc(n,sizeof(double));
45
46     matriz2 = (double**)calloc(n,sizeof(double*));
47     for (i = 0; i < n; i++)
48         matriz2[i] = (double*)calloc(n,sizeof(double));
49
50     matriz3 = (double**)calloc(n,sizeof(double*));
51     for (i = 0; i < n; i++)
52         matriz3[i] = (double*)calloc(n,sizeof(double));
53
54     matriz4 = (double**)calloc(n,sizeof(double*));
55     for (i = 0; i < n; i++)
56         matriz4[i] = (double*)calloc(n,sizeof(double));
57
58     suma = (double**)calloc(n,sizeof(double*));
59     for (i = 0; i < n; i++)
60         suma[i] = (double*)calloc(n,sizeof(double));
61
62     mul = (double**)calloc(n,sizeof(double*));
63     for (i = 0; i < n; i++)
64         mul[i] = (double*)calloc(n,sizeof(double));
65
66     inv1 = (double**)calloc(n,sizeof(double*));
67     for (i = 0; i < n; i++)
68         inv1[i] = (double*)calloc(n,sizeof(double));
69
70     inv2 = (double**)calloc(n,sizeof(double*));
71     for (i = 0; i < n; i++)
72         inv2[i] = (double*)calloc(n,sizeof(double));
73
74     //-----Revisamos que se haya creado la mem compartida-----
75     if((shmid = shmget(llave, TAM_MEM, IPC_CREAT | 0666)) < 0)
76     {
77         perror("Error al obtener la memoria compartida: shmget");
78         exit(-1);
79     }
80
81     if((shm = shmat(shmid, NULL, 0)) == (int*)-1)
82     {
83         perror("Error al lanzar la memoria compartida: shmat");
84         exit(-1);
85     }
86
87     //-----Creamos los procesos-----
88     if((pid = fork()) == 0)
89     {
90         if((pid = fork()) == 0)
91         {
92             //-----PROCESO HIJO: SUMA
93             sleep(11);
94             printf("-----\n");
95             printf("Proceso HIJO leyendo matriz 3\n");
96             leer(matriz3, n, 789);

```

```
97         sleep(4);
98
99         printf("Proceso HIJO leeyendo matriz 4\n");
100         leer(matriz4, n, 890);
101
102         printf("Proceso HIJO sumando matrices\n");
103         sumar(matriz3, matriz4, suma, n);
104         sleep(2);
105
106         printf("Proceso HIJO escribiendo la suma\n");
107         escribir(suma, n, 654);
108         exit(0);
109     }
110     else
111     {
112         //-----PROCESO PADRE: MULTIPLICACION
113         srand(getpid());
114         printf("-----\n");
115         sleep(1);
116
117         printf("Proceso PADRE leeyendo matriz 1\n");
118         leer(matriz1, n, 456);
119         sleep(3);
120
121         printf("Proceso PADRE leeyendo matriz 2\n");
122         leer(matriz2, n, 678);
123         sleep(2);
124
125         printf("Proceso PADRE multiplicando matrices\n");
126         multiplicar(matriz1, matriz2, mul, n);
127
128         printf("Proceso PADRE escribiendo la multiplicacion\n");
129         escribir(mul, n, 654);
130         sleep(2);
131
132         printf("Proceso PADRE creando matriz 3\n");
133         llenar(matriz3, n);
134         imprimir(matriz3, n);
135
136         printf("Proceso PADRE escribiendo matriz 3\n");
137         escribir(matriz3, n, 789);
138         sleep(2);
139
140         printf("\nProceso PADRE creando matriz 4\n");
141         llenar(matriz4, n);
142         imprimir(matriz4, n);
143
144         printf("Proceso PADRE escribiendo matriz 4\n");
145         escribir(matriz4, n, 890);
146         exit(0);
147     }
148 }
149 else
150 {
151     //-----PROCESO ABUELO: Crea matrices y lee inversas
152     srand(getpid());
153     printf("-----\n");
154     printf("Proceso ABUELO creando matriz 1\n");
155     llenar(matriz1, n);
156     imprimir(matriz1, n);
157
158     printf("\nProceso ABUELO creando matriz 2\n");
159     llenar(matriz2, n);
160     imprimir(matriz2, n);
161
162     printf("Proceso ABUELO escribiendo matriz 1\n");
```

```
163     escribir(matriz1, n, 456);
164     sleep(1);
165
166     printf("Proceso ABUELO escribiendo matriz 2\n");
167     escribir(matriz2, n, 678);
168     sleep(3.5);
169
170     printf("Proceso ABUELO leeyendo multiplicacion\n");
171     leer(mul, n, 654);
172     sleep(12);
173
174     printf("Proceso ABUELO leeyendo suma\n");
175     leer(suma, n, 654);
176     sleep(1);
177
178     printf("Suma realizada por el Proceso HIJO:\n");
179     imprimir(suma, n);
180
181     printf("Escribiendo archivo TXT de inversa suma...\n");
182     inversa(suma, inv1, n);
183     crearArchivo(inv1, n, "/inv_suma.txt", path);
184     printf("Listo\n\n");
185
186     printf("Multiplicacion realizada por el Proceso PADRE:\n");
187     imprimir(mul, n);
188
189     printf("Escribiendo archivo TXT de inversa multiplicacion...\n");
190     inversa(mul, inv2, n);
191     crearArchivo(inv2, n, "/inv_mul.txt", path);
192     printf("Listo\n\n");
193     exit(0);
194 }
195 return 0;
196 }
197
198 void escribir(double **matriz, int n, int clave)
199 {
200     int j, i;
201     z = shm;
202     for(i = 0; i < n; i++)
203     {
204         for(j = 0; j < n; j++)
205         {
206             *z++ = matriz[i][j];
207         }
208     }
209
210     *z = clave;
211     while(*shm != '*')
212         sleep(1);
213 }
214
215 void leer(double **matriz, int n, int clave)
216 {
217     int x = 0, y = 0, valor;
218     for(z = shm; *z != clave; z++)
219     {
220         valor = *z;
221         matriz[x][y] = valor;
222
223         if(y == n-1)
224         {
225             x++;
226             y = 0;
227         }
228         else
```

```

229         y++;
230     }
231     *shm = '*';
232 }
233
234 char* leerDirectorio()
235 {
236     char* directorio = (char*)calloc(2000,sizeof(char));
237     printf("Ingresa el nuevo directorio: ");
238     scanf("%s", directorio);
239     return directorio;
240 }

```

2.3.2. Sección Windows

✓ Punto 4: Inversa de la suma y multiplicación de matrices con tuberías

```

1  #include <windows.h>
2  #include <stdio.h>
3  #include <time.h>
4  #include "funciones.h"
5  #define n 10
6
7  char* leerDirectorio();
8  HANDLE proceso(char *name, HANDLE hRead, int nivel);
9  void crearTuberia(HANDLE *hRead, HANDLE *hWrite);
10 void escribir(double **A, HANDLE hWrite);
11 void leer(double **A, HANDLE hRead);
12
13 int main(int argc, char *argv[])
14 {
15     int i, j, k, l, m, o, p;
16     srand(time(NULL));
17     int nivel = 0; // 0-PADRE, 1-HIJO, 2-NIETO
18     HANDLE hRead, hWrite;
19
20     if(argc > 1)
21         sscanf(argv[1], "%d", &nivel);
22     if(nivel == 0) // PADRE
23     {
24         crearTuberia(&hRead, &hWrite);
25         // ----- CREAMOS DOS MATRICES
26         double **matriz1, **matriz2;
27
28         // Inicializa las matrices.
29         matriz1 = (double**)calloc(n,sizeof(double*));
30         for (i = 0; i < n; i++)
31             matriz1[i] = (double*)calloc(n,sizeof(double));
32
33         matriz2 = (double**)calloc(n,sizeof(double*));
34         for (i = 0; i < n; i++)
35             matriz2[i] = (double*)calloc(n,sizeof(double));
36
37         // Llena matriz 1 y matriz 2
38         llenar(matriz1, n);
39         llenar(matriz2, n);
40
41         //----- ENVIA MATRICES A SU HIJO
42         printf(" --> Soy el PADRE y envio la matriz 1 a mi HIJO.\n\n");
43         escribir(matriz1, hWrite);
44         printf(" --> Soy el PADRE y envio la matriz 2 a mi HIJO.\n\n");
45         escribir(matriz2, hWrite);
46         WriteFile(hWrite, &hWrite, sizeof(HANDLE), NULL, NULL);

```

```

47     HANDLE hProc = proceso(argv[0], hRead, 1);
48     WaitForSingleObject(hProc, INFINITE);
49
50     // ----- CREAM MATRIZ
51     double **mul;
52
53     // Inicializa las matrices.
54     mul = (double**)calloc(n,sizeof(double*));
55     for (i = 0; i < n; i++)
56         mul[i] = (double*)calloc(n,sizeof(double));
57
58     // IMPRIMO MATRICES
59     printf("MATRIZ 1\n"); imprimir(matriz1, n);
60     printf("MATRIZ 2\n"); imprimir(matriz2, n);
61
62     // ----- RECIBE MATRIZ MULTIPLICACION
63     leer(mul, hRead);
64     printf(" --> Soy el PADRE y recibi la multiplicacion de mi HIJO.\n");
65     printf("RESULTADO DE LA MULTIPLICACION MATRIZ 1 Y MATRIZ 2\n"); imprimir(mul, n);
66
67     //----- CREAM MATRIZ
68     double **suma;
69
70     // Inicializa las matrices.
71     suma = (double**)calloc(n,sizeof(double*));
72     for (i = 0; i < n; i++)
73         suma[i] = (double*)calloc(n,sizeof(double));
74
75     //----- RECIBE MATRIZ SUMA
76
77     printf(" --> Soy el PADRE y recibi la suma de mi NIETO.\n");
78     leer(suma, hRead);
79
80     // IMPRIME ELEMENTOS DE LA SUMA
81     printf("RESULTADO DE LA SUMA DE LA MATRIZ A Y MATRIZ B\n"); imprimir(suma, n);
82
83     //----- CREAM DOS MATRICES
84     double **invMul, **invSuma;
85
86     // Inicializa las matrices.
87     invMul = (double**)calloc(n,sizeof(double*));
88     for (i = 0; i < n; i++)
89         invMul[i] = (double*)calloc(n,sizeof(double));
90
91     invSuma = (double**)calloc(n,sizeof(double*));
92     for (i = 0; i < n; i++)
93         invSuma[i] = (double*)calloc(n,sizeof(double));
94
95     //----- OBTIENE INVERSAS Y GENERA ARCHIVOS
96     char* path = leerDirectorio();//Obtenemos el directorio desde la entrada de teclado
97     if(!CreateDirectory(path, NULL))
98     {
99         perror(path);
100         exit(-1);
101     }
102
103     printf(" --> Soy el PADRE y estoy sacando inversa y haciendo archivos\n");
104     //Revisamos si la matriz tiene inversa
105     if(inversa(mul, invMul, n) != 0)
106         crearArchivo(invMul, path, "inversa_Mul.txt");
107
108     if(inversa(suma, invSuma, n) != 0)
109         crearArchivo(invSuma, path, "inversa_Suma.txt");
110
111     printf("INVERSA SUMA\n"); imprimir(invSuma, n);
112     printf("INVERSA MULTIPLICACION\n"); imprimir(invMul, n);

```



```

113     }
114     else if(nivel == 1)    // HIJO
115     {
116         hRead = GetStdHandle(STD_INPUT_HANDLE);
117
118         //----- CREAMOS DOS MATRICES
119         double **A, **B;
120
121         // Inicializa las matrices.
122         A = (double**)calloc(n,sizeof(double*));
123         for (i = 0; i < n; i++)
124             A[i] = (double*)calloc(n,sizeof(double));
125
126         B = (double**)calloc(n,sizeof(double*));
127         for (i = 0; i < n; i++)
128             B[i] = (double*)calloc(n,sizeof(double));
129
130         //----- RECIBIR MATRICES DEL PADRE
131         printf(" --> Soy el HIJO y recibí matriz 1 de mi PADRE.\n");
132         leer(A, hRead);
133         printf(" --> Soy el HIJO y recibí matriz 2 de mi PADRE.\n");
134         leer(B, hRead);
135
136         //----- CREAMOS MATRIZ
137         double **AB;
138
139         // Inicializa las matrices.
140         AB = (double**)calloc(n,sizeof(double*));
141         for (i = 0; i < n; i++)
142             AB[i] = (double*)calloc(n,sizeof(double));
143
144         //----- CALCULA MULTIPLICACION
145         printf("Soy el HIJO y estoy multiplicando matriz 1 y 2\n");
146         multiplicar(A, B, AB, n);
147         imprimir(AB, n);
148         ReadFile(hRead, &hWrite, sizeof(HANDLE), NULL, NULL);
149         escribir(AB, hWrite);
150
151         // ----- CREA TUBERÍA
152         HANDLE hRead2, hWrite2;
153         crearTuberia(&hRead2, &hWrite2);
154
155         //----- CREAMOS DOS MATRICES
156         double **M1, **M2;
157
158         // Inicializa las matrices.
159         M1 = (double**)calloc(n,sizeof(double*));
160         for (i = 0; i < n; i++)
161             M1[i] = (double*)calloc(n,sizeof(double));
162
163         M2 = (double**)calloc(n,sizeof(double*));
164         for (i = 0; i < n; i++)
165             M2[i] = (double*)calloc(n,sizeof(double));
166
167         // Llena matriz 1 y matriz 2
168         llenar(M1, n);
169         llenar(M2, n);
170
171         //----- ENVIA MATRICES A SU HIJO
172         printf(" --> Soy el HIJO y envío matriz A al NIETO.\n\n");
173         escribir(M1, hWrite2);
174         printf(" --> Soy el HIJO y envío matriz B al NIETO.\n\n");
175         escribir(M2, hWrite2);
176         printf("MATRIZ A\n"); imprimir(M1, n);
177         printf("MATRIZ B\n"); imprimir(M2, n);
178

```

```

179     WriteFile(hWrite2, &hWrite, sizeof(HANDLE), NULL, NULL);
180     HANDLE hProc = proceso(argv[0], hRead2, 2);
181     WaitForSingleObject(hProc, INFINITE);
182 }
183 else if(nivel == 2) // NIETO
184 {
185     hRead = GetStdHandle(STD_INPUT_HANDLE);
186
187     //----- CREAMOS DOS MATRICES
188     double **sumaA, **sumaB;
189
190     // Inicializa las matrices.
191     sumaA = (double**)calloc(n,sizeof(double*));
192     for (i = 0; i < n; i++)
193         sumaA[i] = (double*)calloc(n,sizeof(double));
194
195     sumaB = (double**)calloc(n,sizeof(double*));
196     for (i = 0; i < n; i++)
197         sumaB[i] = (double*)calloc(n,sizeof(double));
198
199     //----- RECIBIR MATRICES DEL PADRE
200     printf(" --> Soy el NIETO y recibo matriz A del HIJO.\n");
201     leer(sumaA, hRead);
202     printf(" --> Soy el NIETO y recibo matriz B del HIJO.\n");
203     leer(sumaB, hRead);
204     ReadFile(hRead, &hWrite, sizeof(HANDLE), NULL, NULL);
205
206     //----- CREAMOS MATRIZ
207     double **sumaAB;
208
209     // Inicializa las matrices.
210     sumaAB = (double**)calloc(n,sizeof(double*));
211     for (i = 0; i < n; i++)
212         sumaAB[i] = (double*)calloc(n,sizeof(double));
213
214     //----- CALCULA LA SUMA
215     printf(" --> Soy el NIETO y estoy sumando matriz A y B\n");
216     sumar(sumaA, sumaB, sumaAB, n);
217     escribir(sumaAB, hWrite);
218 }
219 return 0;
220 }
221
222 // -----
223 // FUNCIONES
224 // -----
225
226 char* leerDirectorio()
227 {
228     char* directorio = (char*)calloc(2000,sizeof(char));
229     printf("Ingrese el nuevo directorio: ");
230     scanf("%s", directorio);
231     return directorio;
232 }
233
234 HANDLE proceso(char *name, HANDLE hRead, int nivel)
235 {
236     STARTUPINFO si;
237     PROCESS_INFORMATION pi;
238     ZeroMemory(&pi, sizeof(pi));
239     ZeroMemory(&si, sizeof(si));
240     GetStartupInfo(&si);
241     si.hStdInput = hRead;
242     si.hStdError = GetStdHandle(STD_ERROR_HANDLE);
243     si.hStdOutput = GetStdHandle(STD_OUTPUT_HANDLE);
244     si.dwFlags = STARTF_USESTDHANDLES;

```

```

245     si.cb = sizeof(si);
246     char args[100];
247     sprintf(args, "%s %d", name, nivel);
248     CreateProcess(NULL, args, NULL, NULL, TRUE, 0, NULL, NULL, &si, &pi);
249     return pi.hProcess;
250 }
251
252 void crearTuberia(HANDLE *hRead, HANDLE *hWrite)
253 {
254     SECURITY_ATTRIBUTES pipeSeg = {sizeof(SECURITY_ATTRIBUTES), NULL, TRUE};
255     CreatePipe(hRead, hWrite, &pipeSeg, 0);
256 }
257
258 void escribir(double **A, HANDLE hWrite)
259 {
260     int i, j;
261     for(i = 0; i < n; i++)
262     {
263         for(j = 0; j < n; j++)
264             WriteFile(hWrite, &A[i][j], sizeof(double), NULL, NULL);
265     }
266 }
267
268 void leer(double **A, HANDLE hRead)
269 {
270     int i, j;
271     for(i = 0; i < n; i++)
272     {
273         for(j = 0; j < n; j++)
274             ReadFile(hRead, &A[i][j], sizeof(double), NULL, NULL);
275     }
276 }

```

✓ Punto 7: Inversa de la suma y multiplicación de matrices con memoria compartida

PROGRAMA ABUELO: Envía 2 matrices a multiplicar, recibe suma y multiplicación, guarda sus inversas en archivos.

```

1  //Compilar: gcc 7_1.c -o 7
2  //Ejecutar: 7
3
4  #include <stdio.h>
5  #include <stdlib.h>
6  #include <windows.h>
7  #include <string.h>
8  #include "funciones.h"
9  #define TAM_MEM 27
10
11 char* leerDirectorio()
12 {
13     char* directorio = (char*)calloc(2000,sizeof(char));
14     printf("Ingrese el nuevo directorio: ");
15     scanf("%s", directorio);
16     return directorio;
17 }
18
19 int main(void)
20 {
21     // CREAMOS DIRECTORIO
22     char* path = leerDirectorio();//Obtenemos el directorio desde la entrada de teclado
23     if(!CreateDirectory(path, NULL))
24     {
25         perror(path);
26         exit(-1);
27     }

```

```

28 //Para crear el proceso
29 STARTUPINFO si;
30 PROCESS_INFORMATION pi;
31 char *argv[2];
32 ZeroMemory(&si, sizeof(si));
33 si.cb = sizeof(si);
34 ZeroMemory(&pi, sizeof(pi));
35 argv[0] = "C:\\Users\\YaKerTaker\\Google Drive\\5to
    ↳ SEMESTRE\\Sistemas-Operativos\\Practica6\\Windows\\padre";
36 argv[1] = NULL;
37 double **mul, **suma, **matriz1, **matriz2, **inv1, **inv2;
38 char *HP = "HP"; //Padre hijo
39 char *PH = "PH"; //hijo padre
40 char *NP = "NP"; //nieto padre
41 HANDLE hArchMapeoPH, hArchMapeoHP, hArchMapeoNP; //1, hArchMapeo2;
42 int i, j, k, n = 10;
43 int *aPH, *aHP, *aNP;
44 int *shmPH, *shmHP, *shmNP;
45
46 matriz1 = (double**)calloc(n, sizeof(double*));
47 for (i = 0; i < n; i++)
48     matriz1[i] = (double*)calloc(n, sizeof(double));
49
50 matriz2 = (double**)calloc(n, sizeof(double*));
51 for (i = 0; i < n; i++)
52     matriz2[i] = (double*)calloc(n, sizeof(double));
53
54 mul = (double**)calloc(n, sizeof(double*));
55 for (i = 0; i < n; i++)
56     mul[i] = (double*)calloc(n, sizeof(double));
57
58 suma = (double**)calloc(n, sizeof(double*));
59 for (i = 0; i < n; i++)
60     suma[i] = (double*)calloc(n, sizeof(double));
61
62 inv1 = (double**)calloc(n, sizeof(double*));
63 for (i = 0; i < n; i++)
64     inv1[i] = (double*)calloc(n, sizeof(double));
65
66 inv2 = (double**)calloc(n, sizeof(double*));
67 for (i = 0; i < n; i++)
68     inv2[i] = (double*)calloc(n, sizeof(double));
69
70 if (!CreateProcess(NULL, argv[0], NULL, NULL, FALSE, 0, NULL, NULL, &si, &pi))
71 {
72     printf("Fallo al invocar CreateProcess(%i)\n", GetLastError());
73     exit(-1);
74 }
75 srand(GetCurrentProcessId());
76 //WaitForSingleObject(pi.hProcess, INFINITE);
77
78 //MANDA 2 MATRICES A PADRE
79 printf("ABUELO ENVIA 2 MATRICES. ABUELO -> PADRE\n");
80 printf("Matriz 1:\n");
81 llenar(matriz1, n);
82 imprimir(matriz1, n);
83
84 printf("\nMatriz 2:\n");
85 llenar(matriz2, n);
86 imprimir(matriz2, n);
87
88 if ((hArchMapeoPH = CreateFileMapping(INVALID_HANDLE_VALUE, NULL, PAGE_READWRITE, 0, TAM_MEM, PH)) == NULL)
89 {
90     printf("No se mapeo la memoria compartida: (%i)\n", GetLastError());
91     exit(-1);
92 }

```

```

93     if((shmPH = (int *)MapViewOfFile(hArchMapeoPH,FILE_MAP_ALL_ACCESS,0,0,TAM_MEM)) == NULL)
94     {
95         printf("No se creo la memoria compartida: (%i)\n",GetLastError());
96         CloseHandle(hArchMapeoPH);
97         exit(-1);
98     }
99     aPH = shmPH;
100
101     for(i = 0 ; i < 10 ; i++)
102     {
103         for(j = 0 ; j < 10 ; j++)
104         {
105             *aPH = matriz1[i][j];
106             *aPH++;
107         }
108     }
109     for(i = 0 ; i < 10 ; i++)
110     {
111         for(j = 0 ; j < 10 ; j++)
112         {
113             *aPH = *aPH = matriz2[i][j];
114             *aPH++;
115         }
116     }
117     *aPH = 101;
118     while(*shmPH != -1)
119         Sleep(1);
120     UnmapViewOfFile(shmPH);
121     CloseHandle(hArchMapeoPH);
122
123     //RECIBE MATRIZ DEL HIJO
124     printf("ABUELO RECIBE PRODUCTO. ABUELO <- PADRE\n\nProducto de 1 y 2:\n");
125     if((hArchMapeoHP = OpenFileMapping(FILE_MAP_ALL_ACCESS,FALSE,HP)) == NULL)
126     {
127         printf("No se ario archsadsfdfsdfdfdivo de mapeo de la memoria: (%i)\n", GetLastError());
128         exit(-1);
129     }
130     if((shmHP = (int *)MapViewOfFile(hArchMapeoHP,FILE_MAP_ALL_ACCESS,0,0,TAM_MEM)) == NULL)
131     {
132         printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
133         CloseHandle(hArchMapeoHP);
134         exit(-1);
135     }
136     aHP = shmHP;
137     for(i = 0 ; i < 10 ; i++)
138     {
139         for(j = 0 ; j < 10 ; j++)
140         {
141             mul[i][j] = *aHP++;
142             //aHP++;
143         }
144     }
145     *shmHP = -1;
146
147     imprimir(mul, n);
148     printf("\n");
149     UnmapViewOfFile(shmHP);
150     CloseHandle(hArchMapeoHP);
151
152     //RECIBE MATRIZ DEL HIJO
153     printf("ABUELO RECIBE SUMA. ABUELO <- HIJO\n\nSuma de 3 y 4:\n");
154     if((hArchMapeoNP = OpenFileMapping(FILE_MAP_ALL_ACCESS,FALSE,NP)) == NULL)
155     {
156         printf("No se abrio el archivo de mapeo de la memoria: (%i)\n", GetLastError());
157         exit(-1);
158     }

```

```

159     if((shmNP = (int *)MapViewOfFile(hArchMapeoNP,FILE_MAP_ALL_ACCESS,0,0,TAM_MEM)) == NULL)
160     {
161         printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
162         CloseHandle(hArchMapeoNP);
163         exit(-1);
164     }
165     aNP = shmNP;
166     for(i = 0 ; i < 10 ; i++)
167     {
168         for(j = 0 ; j < 10 ; j++)
169         {
170             suma[i][j] = *aNP++;
171         }
172     }
173     *shmNP = -1;
174
175     imprimir(suma, n);
176     printf("\n");
177
178     UnmapViewOfFile(shmNP);
179     CloseHandle(hArchMapeoNP);
180
181     if(inversa(mul, inv1, n) != 0){
182         crearArchivo(inv1, path, "inv_mul.txt");
183         printf("Archivo de la inversa del producto escrito.... inv_mul.txt\n");
184     }
185
186     if(inversa(suma, inv2, n) != 0){
187         crearArchivo(inv2, path, "inv_suma.txt");
188         printf("Archivo de la inversa de la suma escrito.... inv_suma.txt\n");
189     }
190     CloseHandle(pi.hProcess);
191     CloseHandle(pi.hThread);
192     return 0;
193 }

```

PROGRAMA PADRE: Envía 2 matrices a sumar, multiplica matrices recibidas del abuelo y la envía al abuelo.

```

1  //Compilar: gcc 7_2.c -o padre
2
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <windows.h>
6  #include <string.h>
7  #include "funciones.h"
8  #define TAM_MEM 27
9
10 int main(int argc, char *argv[])
11 {
12     Sleep(10);
13     STARTUPINFO siH;
14     PROCESS_INFORMATION piH;
15     char *argvH[2];
16     ZeroMemory(&siH,sizeof(siH));
17     siH.cb = sizeof(siH);
18     ZeroMemory(&piH, sizeof(piH));
19     argvH[0] = "C:\\Users\\YaKerTaker\\Google Drive\\5to
        ↳ SEMESTRE\\Sistemas-Operativos\\Practica6\\Windows\\hijo";
20     argvH[1] = NULL;
21
22     double **matriz1, **matriz2, **matriz3, **matriz4, **producto;
23     int aux, suma;
24     char *PH = "PH"; //padre hijo
25     char *HP = "HP"; //hijo padre

```

```

26     char *HN = "HN"; //hijo nieto
27     HANDLE hArchMapeoPH, hArchMapeoHP, hArchMapeoHN;
28     int i, j, k, shmid, n = 10;
29     int *aPH, *aHP, *aHN;
30     int *shmPH, *shmHP, *shmHN;
31
32     matriz1 = (double**)calloc(n,sizeof(double*));
33     for (i = 0; i < n; i++)
34         matriz1[i] = (double*)calloc(n,sizeof(double));
35
36     matriz2 = (double**)calloc(n,sizeof(double*));
37     for (i = 0; i < n; i++)
38         matriz2[i] = (double*)calloc(n,sizeof(double));
39
40     matriz3 = (double**)calloc(n,sizeof(double*));
41     for (i = 0; i < n; i++)
42         matriz3[i] = (double*)calloc(n,sizeof(double));
43
44     matriz4 = (double**)calloc(n,sizeof(double*));
45     for (i = 0; i < n; i++)
46         matriz4[i] = (double*)calloc(n,sizeof(double));
47
48     producto = (double**)calloc(n,sizeof(double*));
49     for (i = 0; i < n; i++)
50         producto[i] = (double*)calloc(n,sizeof(double));
51
52     if(!CreateProcess(NULL,argvH[0],NULL,NULL,FALSE,0,NULL,NULL,&siH,&piH))
53     {
54         printf("Fallo al invocar CreateProcess(%.i)\n",GetLastError());
55         exit(-1);
56     }
57     srand(GetCurrentProcessId());
58
59     //MANDA MATRIZ A NIETO
60     printf("PADRE ENVIA 2 MATRICES. PADRE -> HIJO\n");
61     if((hArchMapeoHN = CreateFileMapping(INVALID_HANDLE_VALUE,NULL,PAGE_READWRITE,0, TAM_MEM, HN)) == NULL)
62     {
63         printf("No se mapeo la memoria compartida: (%.i)\n",GetLastError());
64         exit(-1);
65     }
66     if((shmHN = (int *)MapViewOfFile(hArchMapeoHN,FILE_MAP_ALL_ACCESS,0,0,TAM_MEM)) == NULL)
67     {
68         printf("No se creo la memoria compartida: (%.i)\n",GetLastError());
69         CloseHandle(hArchMapeoHN);
70         exit(-1);
71     }
72     aHN = shmHN;
73
74     for(i = 0 ; i < 10 ; i++)
75     {
76         for(j = 0 ; j < 10 ; j++)
77         {
78             *aHN = rand()%11;
79             matriz3[i][j] = *aHN;
80             *aHN++;
81         }
82     }
83     for(i = 0 ; i < 10 ; i++)
84     {
85         for(j = 0 ; j < 10 ; j++)
86         {
87             *aHN = rand()%11;
88             matriz4[i][j] = *aHN;
89             *aHN++;
90         }
91     }

```

```

92     *aHN = 101;
93     while(*shmHN != -1)
94         Sleep(1);
95
96     printf("\nMatriz 3:\n");
97     imprimir(matriz3, n);
98
99     printf("\nMatriz 4:\n");
100    imprimir(matriz4, n);
101
102    UnmapViewOfFile(shmHN);
103    CloseHandle(hArchMapeoHN);
104
105    //RECIBE MATRIZ DEL PADRE
106    printf("\nPADRE RECIBE 2 MATRICES. PADRE <- ABUELO\n");
107    if((hArchMapeoPH = OpenFileMapping(FILE_MAP_ALL_ACCESS,FALSE,PH)) == NULL)
108    {
109        printf("No se abrio el archivo de mapeo de la memoria: (%i)\n", GetLastError());
110        exit(-1);
111    }
112    if((shmPH = (int *)MapViewOfFile(hArchMapeoPH,FILE_MAP_ALL_ACCESS,0,0,TAM_MEM)) == NULL)
113    {
114        printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
115        CloseHandle(hArchMapeoPH);
116        exit(-1);
117    }
118    aPH = shmPH;
119    for(i = 0 ; i < 10 ; i++)
120    {
121        for(j = 0 ; j < 10 ; j++)
122        {
123            matriz1[i][j] = *aPH;
124            aPH++;
125        }
126    }
127    for(i = 0 ; i < 10 ; i++)
128    {
129        for(j = 0 ; j < 10 ; j++)
130        {
131            matriz2[i][j] = *aPH;
132            aPH++;
133        }
134    }
135    *shmPH = -1;
136
137    UnmapViewOfFile(shmPH);
138    CloseHandle(hArchMapeoPH);
139
140    //HACE EL PRODUCTO
141    printf("PADRE REALIZA PRODUCTO DE 1 Y 2.\n");
142    multiplicar(matriz1, matriz2, producto, n);
143
144
145    //MANDA MATRIZ AL ABUELO
146    printf("PADRE ENVIA PRODUCTO AL ABUELO. PADRE -> ABUELO. HIJO.\n");
147    if((hArchMapeoHP = CreateFileMapping(INVALID_HANDLE_VALUE,NULL,PAGE_READWRITE,0, TAM_MEM,HP)) == NULL)
148    {
149        printf("No se mapeo la memoria compartida: (%i)\n",GetLastError());
150        exit(-1);
151    }
152    if((shmHP = (int *)MapViewOfFile(hArchMapeoHP,FILE_MAP_ALL_ACCESS,0,0,TAM_MEM)) == NULL)
153    {
154        printf("No se creo la memoria compartida: (%i)\n",GetLastError());
155        CloseHandle(hArchMapeoHP);
156        exit(-1);
157    }

```



```

158     aHP = shmHP;
159     for(i = 0 ; i < 10 ; i++)
160     {
161         for(j = 0 ; j < 10 ; j++)
162         {
163             *aHP = producto[i][j];
164             *aHP++;
165         }
166     }
167
168     *aHP = 101;
169     while(*shmHP != -1)
170         Sleep(1);
171     UnmapViewOfFile(shmHP);
172     CloseHandle(hArchMapeoHP);
173     CloseHandle(piH.hProcess);
174     CloseHandle(piH.hThread);
175     exit(0); //break;
176 }

```

PROGRAMA HIJO: Recibe 2 matrices a sumar, las suma y envía al abuelo.

```

1  //Compilar: gcc 7_3.c -o hijo
2
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <windows.h>
6  #include <string.h>
7  #include "funciones.h"
8  #define TAM_MEM 27
9
10 int main(int argc, char *argv[])
11 {
12     int i, j, k, shmid, n = 10;
13     double **matriz3, **matriz4, **suma;
14     matriz3 = (double**)calloc(n,sizeof(double*));
15     for (i = 0; i < n; i++)
16         matriz3[i] = (double*)calloc(n,sizeof(double));
17
18     matriz4 = (double**)calloc(n,sizeof(double*));
19     for (i = 0; i < n; i++)
20         matriz4[i] = (double*)calloc(n,sizeof(double));
21
22     suma = (double**)calloc(n,sizeof(double*));
23     for (i = 0; i < n; i++)
24         suma[i] = (double*)calloc(n,sizeof(double));
25
26     char *NP = "NP";
27     char *HN = "HN";
28     HANDLE hArchMapeoNP, hArchMapeoHN;
29
30     int *aHN, *aNP;
31     int *shmNP, *shmHN;
32     //RECIBE MATRIZ DEL PADRE
33     printf("\nHIJO RECIBE 2 MATRICES. HIJO <- PADRE\n");
34     if((hArchMapeoHN = OpenFileMapping(FILE_MAP_ALL_ACCESS,FALSE,HN)) == NULL)
35     {
36         printf("No se abrio el archivo de mapeo de la memoria: (%i)\n", GetLastError());
37         exit(-1);
38     }
39     if((shmHN = (int *)MapViewOfFile(hArchMapeoHN,FILE_MAP_ALL_ACCESS,0,0,TAM_MEM)) == NULL)
40     {
41         printf("No se accedio a la memoria compartida: (%i)\n", GetLastError());
42         CloseHandle(hArchMapeoHN);
43         exit(-1);

```

```
44     }
45     aHN = shmHN;
46     for(i = 0 ; i < 10 ; i++)
47     {
48         for(j = 0 ; j < 10 ; j++)
49         {
50             matriz3[i][j] = *aHN;
51             aHN++;
52         }
53     }
54     for(i = 0 ; i < 10 ; i++)
55     {
56         for(j = 0 ; j < 10 ; j++)
57         {
58             matriz4[i][j] = *aHN;
59             aHN++;
60         }
61     }
62     *shmHN = -1;
63     UnmapViewOfFile(shmHN);
64     CloseHandle(hArchMapeoHN);
65     printf("HIJO REALIZA LA SUMA DE 3 Y 4.\n");
66     sumar(matriz3, matriz4, suma, n);
67
68     //MANDA MATRIZ AL ABUELO
69     printf("HIJO ENVIA SUMA AL ABUELO. HIJO -> ABUELO\n");
70     if((hArchMapeoNP = CreateFileMapping(INVALID_HANDLE_VALUE, NULL, PAGE_READWRITE, 0, TAM_MEM, NP)) == NULL)
71     {
72         printf("No se mapeo la memoria compartida: (%i)\n", GetLastError());
73         exit(-1);
74     }
75     if((shmNP = (int *)MapViewOfFile(hArchMapeoNP, FILE_MAP_ALL_ACCESS, 0, 0, TAM_MEM)) == NULL)
76     {
77         printf("No se creo la memoria compartida: (%i)\n", GetLastError());
78         CloseHandle(hArchMapeoNP);
79         exit(-1);
80     }
81     aNP = shmNP;
82     for(i = 0 ; i < 10 ; i++)
83     {
84         for(j = 0 ; j < 10 ; j++)
85         {
86             *aNP = suma[i][j];
87             *aNP++;
88         }
89     }
90
91     *aNP = 101;
92     while(*shmNP != -1)
93         Sleep(1);
94     UnmapViewOfFile(shmNP);
95     CloseHandle(hArchMapeoNP);
96
97     exit(0);
98 }
```

2.4. Pantallas de ejecución de los programas desarrollados

2.4.1. Sección Linux:

✓ Punto 2: Tuberías en Linux

```
enrike@enrike:~/Escritorio$ gcc 2.c -o 2
2.c: In function 'main':
2.c:23:3: warning: implicit declaration of function 'gets'; did you mean 'fgets'
? [-Wimplicit-function-declaration]
   gets(buf);
   ^~~~~
   fgets
/tmp/ccf0CsD7.o: En la función 'main':
2.c:(.text+0x77): aviso: the 'gets' function is dangerous and should not be used
.
enrike@enrike:~/Escritorio$ ./2
HolaMundo
Se recibio: HolaMundo
cadena
Se recibio: cadena
prueba
Se recibio: prueba
nueva
Se recibio: nueva
^C
enrike@enrike:~/Escritorio$
```

✓ Punto 4: Multiplicación e inversa de matrices con Tuberías

```
enrike@enrike:~/Escritorio$ gcc 4.c -o 4
enrike@enrike:~/Escritorio$ ./4
Ingrese el nuevo directorio: resultados
MATRIZ A
3.000  9.000  7.000  5.000  10.000  8.000  7.000  2.000  0.000  0.000
2.000  7.000  8.000  2.000  1.000  9.000  3.000  6.000  1.000  2.000
6.000  3.000  7.000  7.000  10.000  2.000  7.000  3.000  1.000  7.000
0.000  3.000  3.000  8.000  6.000  3.000  5.000  1.000  3.000  3.000
1.000  3.000  0.000  9.000  5.000  1.000  6.000  6.000  6.000  5.000
8.000  1.000  8.000  2.000  7.000  7.000  5.000  1.000  0.000  6.000
7.000  0.000  7.000  10.000  6.000  3.000  2.000  0.000  4.000  5.000
2.000  3.000  9.000  2.000  1.000  1.000  1.000  5.000  8.000  7.000
10.000  3.000  7.000  7.000  6.000  3.000  2.000  0.000  2.000  0.000
4.000  9.000  9.000  10.000  9.000  5.000  2.000  9.000  3.000  4.000

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

```

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

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

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

Multiplicacion realizada por el Proceso PADRE:
152.000 265.000 370.000 285.000 225.000 353.000 236.000 180.000 245.000 201.000
140.000 201.000 280.000 256.000 190.000 253.000 234.000 154.000 227.000 225.000
174.000 231.000 333.000 233.000 226.000 396.000 270.000 205.000 308.000 215.000
106.000 175.000 214.000 173.000 147.000 263.000 149.000 131.000 182.000 108.000
146.000 188.000 240.000 211.000 172.000 314.000 198.000 173.000 262.000 137.000
123.000 206.000 277.000 213.000 242.000 313.000 242.000 198.000 281.000 220.000
134.000 226.000 242.000 223.000 188.000 305.000 244.000 172.000 257.000 168.000
130.000 196.000 244.000 145.000 161.000 230.000 257.000 189.000 223.000 193.000
119.000 215.000 258.000 240.000 166.000 255.000 237.000 155.000 225.000 161.000
255.000 319.000 416.000 353.000 217.000 436.000 361.000 185.000 333.000 283.000

Escribiendo archivo TXT de inversa multiplicacion....Listo

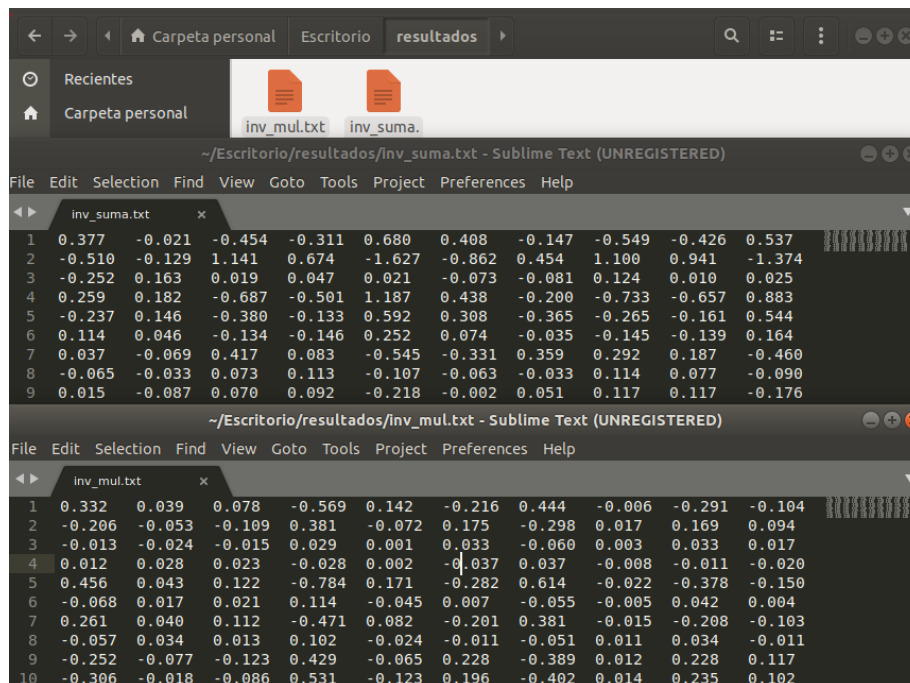
Suma realizada por el Proceso HIJO:
7.000 7.000 10.000 9.000 9.000 16.000 10.000 12.000 8.000 13.000
11.000 11.000 18.000 12.000 8.000 11.000 7.000 0.000 9.000 15.000
146.000 188.000 240.000 211.000 172.000 314.000 198.000 173.000 262.000 137.000
123.000 206.000 277.000 213.000 242.000 313.000 242.000 198.000 281.000 220.000
134.000 226.000 242.000 223.000 188.000 305.000 244.000 172.000 257.000 168.000
130.000 196.000 244.000 145.000 161.000 230.000 257.000 189.000 223.000 193.000
119.000 215.000 258.000 240.000 166.000 255.000 237.000 155.000 225.000 161.000
255.000 319.000 416.000 353.000 217.000 436.000 361.000 185.000 333.000 283.000

Escribiendo archivo TXT de inversa multiplicacion....Listo

Suma realizada por el Proceso HIJO:
7.000 7.000 10.000 9.000 9.000 16.000 10.000 12.000 8.000 13.000
11.000 11.000 18.000 12.000 8.000 11.000 7.000 0.000 9.000 15.000
20.000 10.000 12.000 6.000 15.000 9.000 17.000 16.000 6.000 9.000
10.000 11.000 14.000 18.000 9.000 0.000 10.000 16.000 11.000 16.000
7.000 9.000 5.000 12.000 9.000 11.000 8.000 15.000 13.000 7.000
17.000 9.000 15.000 8.000 13.000 7.000 14.000 8.000 19.000 9.000
3.000 7.000 8.000 16.000 11.000 2.000 14.000 9.000 16.000 12.000
3.000 1.000 9.000 5.000 11.000 17.000 16.000 20.000 17.000 7.000
1.000 10.000 3.000 14.000 14.000 17.000 9.000 6.000 10.000 15.000
3.000 3.000 8.000 9.000 16.000 9.000 20.000 17.000 5.000 15.000

Escribiendo archivo TXT de inversa suma....Listo
enrike@enrike:~/Escritorio$

```



✓ Punto 5: Memoria compartida en Linux

```
enrike@enrike:~/Escritorio$ gcc 5_cliente.c -o cli
enrike@enrike:~/Escritorio$ ./cli
abcdefghijklmnopqrstuvwxyz
enrike@enrike:~/Escritorio$
```

```
enrike@enrike:~/Escritorio$ gcc 5_servidor.c -o serv
enrike@enrike:~/Escritorio$ ./serv
enrike@enrike:~/Escritorio$
```

✓ Punto 7: Inversa de la suma y multiplicación de matrices con memoria compartida

```
enrike@enrike:~/Escritorio$ gcc 7.c -o 7
enrike@enrike:~/Escritorio$ ./7
Ingrese el nuevo directorio: resultados
-----
Proceso ABUELO creando matriz 1
1.000 3.000 10.000 4.000 5.000 6.000 10.000 2.000 6.000 5.000
9.000 0.000 6.000 3.000 6.000 9.000 4.000 9.000 3.000 2.000
0.000 7.000 1.000 0.000 0.000 10.000 4.000 0.000 9.000 7.000
1.000 10.000 10.000 9.000 2.000 2.000 5.000 1.000 5.000 9.000
4.000 1.000 10.000 10.000 2.000 3.000 6.000 6.000 1.000 9.000
8.000 10.000 3.000 7.000 0.000 3.000 4.000 4.000 1.000 3.000
9.000 1.000 0.000 7.000 8.000 2.000 7.000 0.000 1.000 1.000
10.000 4.000 2.000 7.000 1.000 2.000 8.000 6.000 8.000 9.000
2.000 4.000 9.000 6.000 9.000 7.000 7.000 1.000 0.000 9.000
4.000 7.000 8.000 2.000 3.000 5.000 3.000 0.000 6.000 2.000

Proceso ABUELO creando matriz 2
10.000 3.000 6.000 0.000 8.000 6.000 2.000 5.000 1.000 9.000
1.000 3.000 0.000 8.000 9.000 9.000 4.000 4.000 10.000 2.000
2.000 1.000 10.000 8.000 4.000 0.000 2.000 5.000 9.000 6.000
7.000 7.000 9.000 1.000 7.000 4.000 7.000 7.000 9.000 6.000
3.000 0.000 9.000 3.000 8.000 6.000 2.000 0.000 10.000 10.000
2.000 10.000 10.000 10.000 5.000 1.000 0.000 5.000 6.000 7.000
```

```

10.000  7.000  3.000  3.000  5.000  4.000  9.000  5.000  0.000  1.000

Proceso ABUELO escribiendo matriz 1
-----
Proceso PADRE leyendo matriz 1
Proceso ABUELO escribiendo matriz 2
Proceso PADRE leyendo matriz 2
Proceso PADRE multiplicando matrices
Proceso PADRE escribiendo la multiplicacion
Proceso ABUELO leyendo multiplicacion
Proceso PADRE creando matriz 3
5.000  6.000  1.000  9.000  7.000  4.000  5.000  3.000  4.000  7.000
6.000  3.000  10.000  1.000  4.000  10.000  9.000  8.000  1.000  3.000
2.000  7.000  9.000  8.000  7.000  2.000  9.000  9.000  6.000  3.000
6.000  9.000  7.000  8.000  8.000  3.000  10.000  0.000  6.000  3.000
5.000  0.000  4.000  5.000  10.000  8.000  4.000  8.000  3.000  5.000
9.000  3.000  1.000  6.000  0.000  7.000  8.000  9.000  5.000  1.000
10.000  10.000  0.000  6.000  5.000  6.000  8.000  4.000  6.000  3.000
5.000  1.000  1.000  9.000  6.000  0.000  5.000  8.000  7.000  8.000
2.000  5.000  10.000  3.000  9.000  10.000  8.000  7.000  7.000  3.000
6.000  4.000  0.000  6.000  9.000  3.000  10.000  6.000  7.000  6.000

Proceso PADRE escribiendo matriz 3
-----
Proceso PADRE escribiendo matriz 3
-----
Proceso HIJO leyendo matriz 3

Proceso PADRE creando matriz 4
7.000  10.000  5.000  7.000  6.000  9.000  7.000  0.000  6.000  3.000
7.000  8.000  7.000  6.000  9.000  5.000  3.000  7.000  10.000  8.000
8.000  6.000  0.000  6.000  10.000  7.000  9.000  10.000  0.000  5.000
3.000  7.000  2.000  6.000  1.000  8.000  4.000  9.000  7.000  8.000
1.000  3.000  3.000  8.000  7.000  10.000  1.000  8.000  4.000  0.000
6.000  1.000  4.000  4.000  7.000  4.000  0.000  3.000  1.000  0.000
6.000  4.000  5.000  8.000  8.000  7.000  6.000  10.000  5.000  2.000
7.000  4.000  3.000  10.000  2.000  8.000  7.000  1.000  5.000  1.000
1.000  0.000  0.000  4.000  2.000  8.000  8.000  2.000  9.000  7.000
2.000  5.000  9.000  8.000  2.000  6.000  4.000  6.000  5.000  9.000

Proceso PADRE escribiendo matriz 4
Proceso HIJO leyendo matriz 4
Proceso HIJO sumando matrices
Proceso HIJO escribiendo la suma
Proceso ABUELO leyendo suma
Suma realizada por el Proceso HIJO:
12.000  16.000  6.000  16.000  13.000  13.000  12.000  3.000  10.000  10.000

Proceso HIJO escribiendo la suma
Proceso ABUELO leyendo suma
Suma realizada por el Proceso HIJO:
12.000  16.000  6.000  16.000  13.000  13.000  12.000  3.000  10.000  10.000
13.000  11.000  17.000  7.000  13.000  15.000  12.000  15.000  11.000  11.000
10.000  13.000  9.000  14.000  17.000  9.000  18.000  19.000  6.000  8.000
9.000  16.000  9.000  14.000  9.000  11.000  14.000  9.000  13.000  11.000
6.000  3.000  7.000  13.000  17.000  18.000  5.000  16.000  7.000  5.000
15.000  4.000  5.000  10.000  7.000  11.000  8.000  12.000  6.000  1.000
16.000  14.000  5.000  14.000  13.000  13.000  14.000  14.000  11.000  5.000
12.000  5.000  4.000  19.000  8.000  8.000  12.000  9.000  12.000  9.000
3.000  5.000  10.000  7.000  11.000  18.000  16.000  9.000  16.000  10.000
8.000  9.000  9.000  14.000  11.000  9.000  14.000  12.000  12.000  15.000

Escribiendo archivo TXT de inversa suma....Listo

Multiplicacion realizada por el Proceso PADRE:
196.000  207.000  306.000  314.000  278.000  209.000  187.000  268.000  327.000  300.000
231.000  235.000  366.000  215.000  337.000  187.000  146.000  282.000  267.000  311.000
166.000  224.000  143.000  271.000  209.000  160.000  164.000  177.000  223.000  151.000
246.000  230.000  274.000  292.000  322.000  240.000  259.000  278.000  362.000  242.000
264.000  241.000  349.000  215.000  310.000  193.000  230.000  315.000  276.000  270.000
204.000  194.000  220.000  188.000  302.000  228.000  169.000  249.000  248.000  218.000
192.000  125.000  233.000  124.000  235.000  204.000  117.000  172.000  203.000  285.000
236.000  260.000  265.000  217.000  250.000  274.000  260.000  220.000  250.000  270.000

```



```

6.000 3.000 7.000 13.000 17.000 18.000 5.000 16.000 7.000 5.000
15.000 4.000 5.000 10.000 7.000 11.000 8.000 12.000 6.000 1.000
16.000 14.000 5.000 14.000 13.000 13.000 14.000 14.000 11.000 5.000
12.000 5.000 4.000 19.000 8.000 8.000 12.000 9.000 12.000 9.000
3.000 5.000 10.000 7.000 11.000 18.000 16.000 9.000 16.000 10.000
8.000 9.000 9.000 14.000 11.000 9.000 14.000 12.000 12.000 15.000

Escribiendo archivo TXT de inversa suma....Listo

Multiplicacion realizada por el Proceso PADRE:
196.000 207.000 306.000 314.000 278.000 209.000 187.000 268.000 327.000 300.000
231.000 235.000 366.000 215.000 337.000 187.000 146.000 282.000 267.000 311.000
166.000 224.000 143.000 271.000 209.000 160.000 164.000 177.000 223.000 151.000
246.000 230.000 274.000 292.000 322.000 240.000 259.000 278.000 362.000 242.000
264.000 241.000 349.000 215.000 310.000 193.000 230.000 315.000 276.000 270.000
204.000 194.000 220.000 188.000 302.000 228.000 169.000 249.000 248.000 218.000
192.000 125.000 233.000 124.000 235.000 204.000 117.000 172.000 203.000 285.000
336.000 268.000 265.000 217.000 360.000 274.000 268.000 328.000 259.000 279.000
225.000 222.000 362.000 290.000 313.000 228.000 197.000 259.000 333.000 328.000
161.000 155.000 214.000 247.000 239.000 168.000 138.000 179.000 281.000 216.000

Escribiendo archivo TXT de inversa multiplicacion....Listo

enrike@enrike:~/Escritorio$

```

```

~/Escritorio/resultados/inv_mul.txt - Sublime Text (UNREGISTERED)
Edit Selection Find View Goto Tools Project Preferences Help

inv_mul.txt x
-0.030 -0.049 0.007 -0.065 0.063 0.020 0.002 -0.005 0.006 0.077
-0.003 0.008 0.021 0.012 0.009 -0.001 0.020 -0.019 -0.018 -0.020
-0.021 -0.030 0.000 -0.038 0.032 0.021 -0.009 -0.005 0.028 0.030
0.000 -0.010 -0.008 -0.021 -0.004 0.003 -0.019 0.015 0.021 0.018
0.017 0.054 -0.020 0.055 -0.074 -0.028 -0.019 0.027 0.006 -0.046
-0.024 -0.053 0.010 -0.058 0.051 0.043 0.001 -0.015 0.025 0.042
0.027 0.051 -0.015 0.074 -0.073 -0.039 -0.008 0.022 0.003 -0.076
0.011 -0.013 0.000 -0.018 0.022 0.014 0.002 -0.006 -0.017 0.012
-0.009 -0.013 0.012 -0.004 0.033 0.007 0.021 -0.024 -0.025 0.020

~/Escritorio/resultados/inv_suma.txt - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

inv_suma.txt x
1 0.033 0.033 -0.033 -0.095 -0.038 0.031 0.051 -0.003 -0.018 0.049
2 -0.007 -0.022 -0.011 0.093 0.009 -0.005 0.031 -0.065 -0.030 0.004
3 -0.011 0.131 0.058 0.058 -0.001 -0.067 -0.111 0.145 -0.007 -0.203
4 0.004 -0.017 0.036 0.092 0.031 0.022 -0.108 0.079 -0.022 -0.086
5 0.026 0.112 0.026 -0.167 0.015 -0.231 0.132 0.131 0.002 -0.104
6 0.053 -0.118 -0.031 0.063 0.022 0.214 -0.101 -0.177 0.032 0.119
7 0.034 -0.056 0.075 -0.019 -0.063 0.093 -0.071 -0.030 0.065 0.006
8 -0.081 -0.045 -0.014 0.058 0.028 0.047 0.032 -0.071 -0.025 0.088
9 -0.094 0.115 -0.056 -0.027 0.008 -0.250 0.190 0.182 0.013 -0.139
10 0.046 -0.082 -0.059 -0.047 -0.008 0.123 -0.026 -0.170 -0.020 0.286

```

2.4.2. Sección Windows:

✓ Punto 3: Tuberías en Windows

```

C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>gcc 3_hijo.c -o hijo
C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>gcc 3.c -o 3
C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>3 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\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>

```

✓ Punto 4: Multiplicación e inversa de matrices con Tuberías

```

C:\Users\UnADM\Documents\ESCUELA\GitHub\Sistemas-Operativos\Practica6\Windows\4>
--> Soy el PADRE y envio la matriz 1 a mi HIJO.

--> Soy el PADRE y envio la matriz 2 a mi HIJO.

--> Soy el HIJO y recibí matriz 1 de mi PADRE.
--> Soy el HIJO y recibí matriz 2 de mi PADRE.
Soy el HIJO y estoy multiplicando matriz 1 y 2
203.000 232.000 193.000 296.000 156.000 234.000 207.000 300.000 180.000 230.000
226.000 160.000 226.000 212.000 191.000 126.000 201.000 192.000 180.000 140.000
275.000 223.000 252.000 204.000 181.000 178.000 264.000 244.000 221.000 219.000
256.000 233.000 225.000 361.000 187.000 223.000 229.000 357.000 236.000 284.000
286.000 279.000 214.000 387.000 305.000 240.000 248.000 327.000 237.000 273.000
382.000 376.000 312.000 409.000 246.000 276.000 382.000 372.000 303.000 286.000
251.000 223.000 193.000 289.000 193.000 153.000 227.000 336.000 286.000 261.000
291.000 248.000 286.000 275.000 204.000 195.000 287.000 268.000 215.000 210.000
274.000 233.000 282.000 267.000 144.000 279.000 255.000 260.000 203.000 176.000
338.000 237.000 278.000 308.000 254.000 215.000 277.000 305.000 290.000 261.000

--> Soy el HIJO y envio matriz A al NIETO.

--> Soy el HIJO y envio matriz B al NIETO.

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

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

--> Soy el NIETO y recibo matriz A del HIJO.
--> Soy el NIETO y recibo matriz B del HIJO.
--> Soy el NIETO y estoy sumando matriz A y B
MATRIZ 1
4.000 4.000 5.000 5.000 1.000 6.000 2.000 5.000 8.000 6.000
0.000 1.000 3.000 3.000 9.000 3.000 8.000 6.000 0.000 2.000
5.000 10.000 2.000 5.000 7.000 5.000 2.000 2.000 4.000 3.000
5.000 4.000 1.000 1.000 1.000 7.000 5.000 10.000 9.000 9.000
5.000 1.000 8.000 0.000 7.000 9.000 9.000 5.000 9.000 2.000
8.000 8.000 10.000 5.000 6.000 4.000 8.000 8.000 2.000 4.000
9.000 4.000 1.000 1.000 5.000 7.000 0.000 10.000 8.000 2.000
2.000 6.000 5.000 5.000 7.000 2.000 8.000 6.000 2.000 4.000
1.000 7.000 6.000 9.000 7.000 6.000 2.000 3.000 0.000 10.000
6.000 7.000 1.000 2.000 10.000 10.000 5.000 5.000 4.000 6.000

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

```


ca. Símbolo del sistema

```
--> Soy el PADRE y recibi la multiplicacion de mi HIJO.
RESULTADO DE LA MULTIPLICACION MATRIZ 1 Y MATRIZ 2
203.000 232.000 193.000 296.000 156.000 234.000 207.000 300.000 180.000 230.000
226.000 160.000 226.000 212.000 191.000 126.000 201.000 192.000 180.000 140.000
275.000 223.000 252.000 204.000 181.000 178.000 264.000 244.000 221.000 219.000
256.000 233.000 225.000 361.000 187.000 223.000 229.000 357.000 236.000 284.000
286.000 279.000 214.000 387.000 305.000 240.000 248.000 327.000 237.000 273.000
382.000 376.000 312.000 409.000 246.000 276.000 382.000 372.000 303.000 286.000
251.000 223.000 193.000 289.000 193.000 153.000 227.000 336.000 286.000 261.000
291.000 248.000 286.000 275.000 204.000 195.000 287.000 268.000 215.000 210.000
274.000 233.000 282.000 267.000 144.000 279.000 255.000 260.000 203.000 176.000
338.000 237.000 278.000 308.000 254.000 215.000 277.000 305.000 290.000 261.000

--> Soy el PADRE y recibi la suma de mi NIETO.
RESULTADO DE LA SUMA DE LA MATRIZ A Y MATRIZ B
11.000 11.000 5.000 13.000 7.000 8.000 10.000 15.000 18.000 12.000
10.000 9.000 12.000 4.000 9.000 6.000 18.000 8.000 4.000 9.000
9.000 20.000 2.000 15.000 9.000 15.000 6.000 6.000 5.000 4.000
7.000 10.000 11.000 4.000 3.000 13.000 13.000 18.000 12.000 13.000
14.000 1.000 16.000 0.000 17.000 10.000 14.000 9.000 18.000 3.000
11.000 12.000 11.000 13.000 9.000 10.000 8.000 10.000 5.000 9.000
17.000 11.000 9.000 11.000 15.000 10.000 9.000 15.000 10.000 9.000
8.000 12.000 12.000 14.000 7.000 3.000 14.000 16.000 11.000 10.000
2.000 10.000 9.000 14.000 15.000 10.000 4.000 13.000 2.000 19.000
10.000 7.000 4.000 9.000 10.000 19.000 6.000 11.000 6.000 7.000

Ingrese el nuevo directorio: Resultados
--> Soy el PADRE y estoy sacando inversa y haciendo archivos
INVERSA SUMA
0.038 0.025 -0.042 -0.010 -0.030 0.085 0.062 -0.073 -0.042 -0.012
-0.027 -0.011 0.080 0.059 0.002 -0.025 0.044 -0.031 0.009 -0.099
-0.043 -0.045 -0.012 0.032 0.036 0.126 -0.037 -0.001 -0.002 -0.045
0.028 -0.008 -0.029 -0.076 -0.009 0.055 -0.067 0.066 -0.010 0.072
-0.032 0.004 0.033 -0.028 0.028 -0.104 0.044 0.022 0.045 -0.005
-0.004 -0.005 0.004 0.012 0.010 0.025 -0.051 -0.018 -0.006 0.059
0.013 0.082 -0.003 -0.041 -0.011 -0.072 -0.044 0.052 -0.013 0.065
-0.080 -0.046 -0.006 0.042 -0.022 -0.100 0.087 0.092 -0.005 0.032
0.059 -0.028 0.017 0.006 0.038 0.004 -0.038 -0.021 0.002 -0.034
0.084 0.048 -0.035 0.006 -0.022 0.056 -0.018 -0.105 0.034 -0.037
```





ca. Símbolo del sistema

```
0.084 0.048 -0.035 0.006 -0.022 0.056 -0.018 -0.105 0.034 -0.037



INVERSA MULTIPLICACION
-0.225 -0.079 0.066 0.089 0.104 -0.037 0.053 0.043 0.107 -0.139
-0.266 0.209 0.325 0.217 0.115 0.057 -0.013 -0.373 0.120 -0.337
-0.106 0.178 0.199 0.128 0.037 0.047 -0.034 -0.268 0.045 -0.179
0.063 0.050 -0.011 -0.006 -0.037 0.028 -0.032 -0.050 -0.036 0.039
0.042 -0.047 -0.061 -0.049 -0.008 -0.021 0.012 0.086 -0.017 0.055
0.102 -0.052 -0.094 -0.075 -0.040 -0.012 -0.003 0.090 -0.038 0.110
0.395 -0.244 -0.418 -0.303 -0.170 -0.053 0.004 0.457 -0.182 0.450
-0.071 -0.184 -0.107 -0.042 0.047 -0.078 0.081 0.254 0.047 0.016
0.081 0.112 0.037 -0.004 -0.046 0.046 -0.039 -0.143 -0.039 0.020
0.022 0.101 0.085 0.044 -0.018 0.039 -0.047 -0.152 -0.023 -0.028
```

C:\Users\UnADM\Documents\ESCUELA\GitHub\Sistemas-Operativos\Practica6\Windows\4>

Documentos > ESCUELA > GitHub > Sistemas-Operativos > Practica6 > Windows > 4

Nombre	Fecha de modifica...	Tipo	Tamaño
 Resultados	25/11/2018 09:32 ...	Carpeta de archivos	
 4.c	25/11/2018 08:29 ...	Archivo C	9 KB
 4	25/11/2018 08:28 ...	Aplicación	31 KB
 funciones.h	17/11/2018 02:26 ...	Archivo H	8 KB

Documentos > ESCUELA > GitHub > Sistemas-Operativos > Practica6 > Windows > 4 > Resultados

Nombre	Fecha de modifica...	Tipo	Tamaño
 inversa_Mul	25/11/2018 09:32 ...	Documento de tex...	1 KB
 inversa_Suma	25/11/2018 09:32 ...	Documento de tex...	1 KB

inversa_Suma.txt

inversa_Mul.txt

1	0.038	0.025	-0.042	-0.010	-0.030	0.085	0.062	-0.073	-0.042	-0.012
2	-0.027	-0.011	0.080	0.059	0.002	-0.025	0.044	-0.031	0.009	-0.099
3	-0.043	-0.045	-0.012	0.032	0.036	0.126	-0.037	-0.001	-0.002	-0.045
4	0.028	-0.008	-0.029	-0.076	-0.009	0.055	-0.067	0.066	-0.010	0.072
5	-0.032	0.004	0.033	-0.028	0.028	-0.104	0.044	0.022	0.045	-0.005
6	-0.004	-0.005	0.004	0.012	0.010	0.025	-0.051	-0.018	-0.006	0.059
7	0.013	0.082	-0.003	-0.041	-0.011	-0.072	-0.044	0.052	-0.013	0.065
8	-0.080	-0.046	-0.006	0.042	-0.022	-0.100	0.087	0.092	-0.005	0.032
9	0.059	-0.028	0.017	0.006	0.038	0.004	-0.038	-0.021	0.002	-0.034
10	0.084	0.048	-0.035	0.006	-0.022	0.056	-0.018	-0.105	0.034	-0.037
11										

inversa_Suma.txt

inversa_Mul.txt

1	-0.225	-0.079	0.066	0.089	0.104	-0.037	0.053	0.043	0.107	-0.139
2	-0.266	0.209	0.325	0.217	0.115	0.057	-0.013	-0.373	0.120	-0.337
3	-0.106	0.178	0.199	0.128	0.037	0.047	-0.034	-0.268	0.045	-0.179
4	0.063	0.050	-0.011	-0.006	-0.037	0.028	-0.032	-0.050	-0.036	0.039
5	0.042	-0.047	-0.061	-0.049	-0.008	-0.021	0.012	0.086	-0.017	0.055
6	0.102	-0.052	-0.094	-0.075	-0.040	-0.012	-0.003	0.090	-0.038	0.110
7	0.395	-0.244	-0.418	-0.303	-0.170	-0.053	0.004	0.457	-0.182	0.450
8	-0.071	-0.184	-0.107	-0.042	0.047	-0.078	0.081	0.254	0.047	0.016
9	0.081	0.112	0.037	-0.004	-0.046	0.046	-0.039	-0.143	-0.039	0.020
10	0.022	0.101	0.085	0.044	-0.018	0.039	-0.047	-0.152	-0.023	-0.028
11										

✓ Punto 6: Memoria compartida en Windows

```

C:\Users\YaKerTaker> gcc 6_servidor.c -o serv
C:\Users\YaKerTaker> serv
C:\Users\YaKerTaker>

C:\Users\YaKerTaker> gcc 6_cliente.c -o cli
C:\Users\YaKerTaker> cli
abcdefghijklmnopqrstuvwxyz
C:\Users\YaKerTaker>

```

✓ Punto 7: Inversa de la suma y multiplicación de matrices con memoria compartida

```
C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>gcc 7_3.c -o hijo
C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>gcc 7_2.c -o padre
C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>gcc 7_1.c -o 7
C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>7
Ingrese el nuevo directorio: resultados
ABUELO ENVIA 2 MATRICES. ABUELO -> PADRE

Matriz 1:
2.000 7.000 3.000 10.000 10.000 8.000 1.000 6.000 1.000 6.000
5.000 9.000 1.000 4.000 7.000 2.000 10.000 1.000 8.000 10.000
0.000 0.000 10.000 7.000 7.000 5.000 8.000 8.000 4.000 1.000
1.000 4.000 2.000 7.000 8.000 4.000 2.000 9.000 6.000 5.000
4.000 7.000 6.000 6.000 2.000 1.000 5.000 0.000 8.000 8.000
1.000 9.000 9.000 1.000 9.000 10.000 3.000 7.000 4.000 4.000
3.000 2.000 8.000 2.000 3.000 0.000 6.000 8.000 0.000 5.000
0.000 5.000 10.000 7.000 6.000 2.000 5.000 10.000 9.000
10.000 2.000 1.000 5.000 5.000 10.000 9.000 1.000 4.000 5.000
9.000 10.000 0.000 1.000 7.000 7.000 2.000 9.000 4.000 9.000

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

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

PADRE ENVIA 2 MATRICES. PADRE -> HIJO

HIJO RECIBE 2 MATRICES. HIJO <- PADRE
HIJO REALIZA LA SUMA DE 3 Y 4.
HIJO ENVIA SUMA AL ABUELO. HIJO -> ABUELO

Matriz 3:
10.000 9.000 8.000 0.000 1.000 2.000 8.000 9.000 10.000 7.000
8.000 4.000 10.000 7.000 8.000 4.000 6.000 8.000 6.000 1.000
0.000 8.000 5.000 0.000 6.000 5.000 7.000 9.000 6.000 9.000
8.000 0.000 7.000 8.000 5.000 1.000 3.000 3.000 6.000 3.000
2.000 7.000 10.000 6.000 8.000 4.000 8.000 3.000 3.000 7.000
4.000 2.000 8.000 8.000 8.000 7.000 4.000 6.000 2.000 9.000
1.000 0.000 6.000 5.000 2.000 6.000 7.000 10.000 8.000 0.000
3.000 10.000 9.000 5.000 9.000 0.000 5.000 10.000 10.000 5.000
5.000 9.000 3.000 9.000 8.000 4.000 9.000 6.000 0.000 0.000
3.000 0.000 10.000 10.000 10.000 7.000 3.000 10.000 3.000 8.000

Matriz 4:
1.000 1.000 2.000 10.000 4.000 7.000 4.000 0.000 2.000 1.000
7.000 6.000 6.000 7.000 7.000 4.000 4.000 0.000 1.000 7.000
4.000 1.000 10.000 5.000 4.000 7.000 6.000 4.000 8.000 3.000
6.000 2.000 5.000 5.000 5.000 0.000 8.000 5.000 6.000 5.000
6.000 1.000 4.000 0.000 5.000 4.000 10.000 5.000 3.000 9.000
9.000 7.000 0.000 6.000 5.000 6.000 0.000 0.000 4.000 7.000
0.000 0.000 5.000 0.000 0.000 9.000 6.000 10.000 2.000 1.000
2.000 5.000 1.000 6.000 8.000 0.000 5.000 5.000 5.000 1.000
4.000 4.000 6.000 9.000 5.000 7.000 1.000 0.000 9.000 10.000
7.000 8.000 3.000 10.000 9.000 2.000 4.000 1.000 0.000 9.000

PADRE RECIBE 2 MATRICES. PADRE <- ABUELO
PADRE REALIZA PRODUCTO DE 1 Y 2.
PADRE ENVIA PRODUCTO AL ABUELO. PADRE -> ABUELO. HIJO.
ABUELO RECIBE PRODUCTO. ABUELO <- PADRE

Producto de 1 y 2:
319.000 315.000 295.000 318.000 224.000 297.000 186.000 139.000 218.000 317.000
388.000 323.000 279.000 351.000 283.000 185.000 177.000 250.000 245.000 283.000
205.000 303.000 315.000 330.000 207.000 230.000 193.000 235.000 206.000 209.000
292.000 307.000 271.000 300.000 225.000 224.000 143.000 155.000 161.000 235.000
322.000 242.000 245.000 314.000 224.000 162.000 115.000 242.000 164.000 256.000
307.000 340.000 274.000 334.000 289.000 260.000 226.000 196.000 260.000 311.000
178.000 207.000 205.000 234.000 187.000 127.000 111.000 160.000 155.000 185.000
362.000 324.000 293.000 375.000 275.000 236.000 149.000 240.000 157.000 303.000
297.000 287.000 321.000 287.000 180.000 177.000 184.000 273.000 270.000 232.000
```

```

307.000 340.000 274.000 334.000 289.000 260.000 226.000 196.000 260.000 311.000
178.000 207.000 205.000 234.000 187.000 127.000 111.000 160.000 155.000 185.000
362.000 324.000 293.000 375.000 275.000 236.000 149.000 240.000 157.000 303.000
297.000 287.000 321.000 287.000 180.000 177.000 184.000 273.000 270.000 232.000
402.000 344.000 281.000 311.000 299.000 206.000 163.000 190.000 258.000 323.000

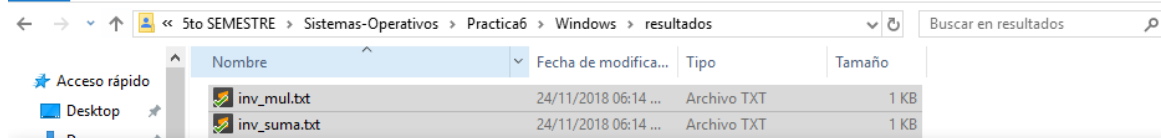
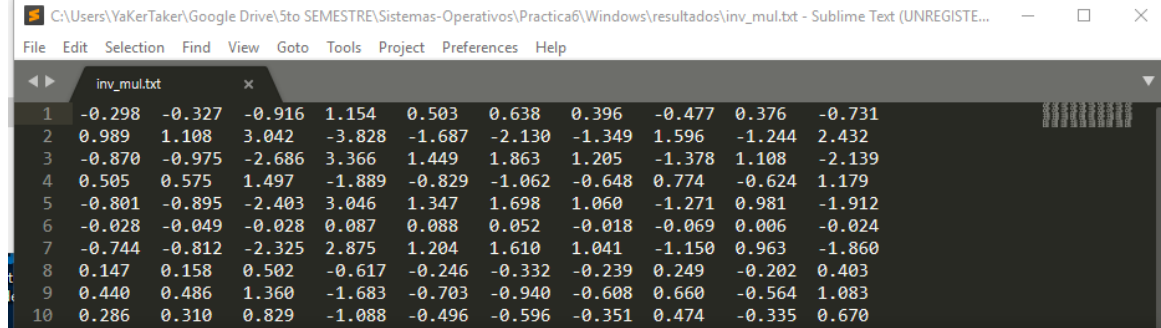
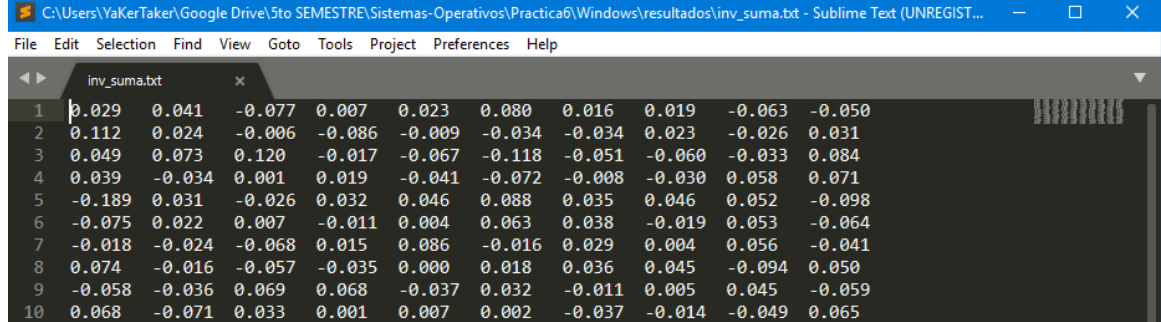
ABUELO RECIBE SUMA. ABUELO <- HIJO

Suma de 3 y 4:
11.000 10.000 10.000 10.000 5.000 9.000 12.000 9.000 12.000 8.000
15.000 10.000 16.000 14.000 15.000 8.000 10.000 8.000 7.000 8.000
4.000 9.000 15.000 5.000 10.000 12.000 13.000 13.000 14.000 12.000
14.000 2.000 12.000 13.000 10.000 1.000 11.000 8.000 12.000 8.000
8.000 8.000 14.000 6.000 13.000 8.000 18.000 8.000 6.000 16.000
13.000 9.000 8.000 14.000 13.000 13.000 4.000 6.000 6.000 16.000
1.000 0.000 11.000 5.000 2.000 15.000 13.000 20.000 10.000 1.000
5.000 15.000 10.000 11.000 17.000 0.000 10.000 15.000 15.000 6.000
9.000 13.000 9.000 18.000 13.000 11.000 10.000 6.000 9.000 10.000
10.000 8.000 13.000 20.000 19.000 9.000 7.000 11.000 3.000 17.000

Archivo de la inversa del producto escrito.... inv_mul.txt
Archivo de la inversa de la suma escrito.... inv_suma.txt

C:\Users\YaKerTaker\Google Drive\5to SEMESTRE\Sistemas-Operativos\Practica6\Windows>

```

3. Observaciones

- ✓ En el código de ejemplo 2 de tuberías en Linux, aparece un warning al momento de compilar el programa, esto debido al uso de la función **gets()**. Sin embargo, la ejecución del programa es la esperada.
- ✓ Para ejecutar los códigos de ejemplo de memoria compartida en ambos sistemas operativos, se deben abrir dos terminales, una para el servidor y otra para el cliente. El servidor debe ejecutarse antes que el cliente.

- ✓ Para todos los programas, al momento de llenar las matrices con números aleatorios utilizando la función **rand()**, primero se inicializó el generador **srand()**, sin embargo, en vez de enviarle como parámetro la clásica función *time(NULL)*, se le envió el número identificador del proceso en donde se encontraba la matriz, esto para evitar que las matrices posean los mismos números aleatorios, debido a la concurrencia en la ejecución de los procesos.

En Linux se envió como parámetro la función **getpid()**, mientras que en Windows se envió la función **GetCurrentProcessId()**:

srand(getpid()), srand(GetCurrentProcessId()).

- ✓ Para la creación de procesos, en Linux se utilizó la creación por copia de exacta de código, mientras que en Windows la creación por sustitución de código, por eso se crearon varios archivos para un mismo programa.
- ✓ Las funciones de operaciones de matrices que se usan en esta práctica, como llenar las matrices con números aleatorios, imprimir las matrices en pantalla, sumar, multiplicar, obtener la inversa y guardar una matriz resultante en un archivo TXT, se encuentran en el archivo llamado "**funciones.h**", que ya se explicó en prácticas pasadas.
- ✓ Para compilar los procesos padre e hijo en los programas de Windows, al inicio del código de cada uno vienen las instrucciones de compilación. Es importante tomar en cuenta esto, debido a que los procesos con mayor jerarquía creados por sustitución de código utilizan rutas o directorios fijos o predeterminados, en donde se encuentran los procesos hijos. En el caso de Linux esto no es necesario.
- ✓ En varios programas, tanto en Linux como Windows, surgió el problema de concurrencia entre procesos, pues algunos intentaban acceder a ciertas variables, por medio de tuberías y memoria dinámica, que aun no se terminaban de calcular. Para solucionar esto, hicimos uso de funciones **sleep()** para detener por un corto periodo la ejecución de los procesos que deseaban acceder a variables que necesitaban un previo procesamiento.
- ✓ En el programa 4 de Linux de tuberías, para enviar y recibir los datos por medio de tuberías utilizando las funciones **read()** y **write()**, se necesitó de dos ciclos anidados para ir mandando posición por posición, de manera individual para evitar errores. En este mismo programa, la creación de procesos se hizo de forma inversa a como la veníamos manejando, trabajando primero sobre el proceso padre y luego sobre los hijos.
- ✓ En el programa 4 de Windows de tuberías, en los resultados se van mostrando, imprimimos cuando llega al PADRE, eso se hace para que notemos que el programa está funcionando de manera correcta y que las tuberías están funcionando. Por lo anterior, se ven comentarios que muestran como funciona el programa, estos sirven para identificar como interactúan.

4. Análisis Crítico

Nuevamente nos valemos de llamadas al sistema para la creación de tuberías y memoria compartida.

En Linux tenemos lo siguiente

Tuberías:

- ✓ Crear una tubería con `pipe()`.
- ✓ Escribir datos con `Write()`.
- ✓ Leer datos con `read()`.

Memoria compartida:

- ✓ Obtiene el identificador con `shmget()`.
- ✓ Adjunta identificador con `shmat()`.

En Windows tenemos lo siguiente **Tuberías:**

- ✓ Crear una tubería con `CreatePipe()`.
- ✓ Escribir datos con `WriteFile()`.
- ✓ Leer datos con `ReadFile()`.

Recordando la información que el profesor nos dio en la clase, se mencionaba que en el caso de las tuberías el Sistema operativo trabaja la información como archivos, la razón es porque las variables e incluso apuntadores no se permiten compartir entre procesos.

Otra característica que se vio en el desarrollo de esta práctica, es que las tuberías no son bidireccionales, eso se nota porque para enviar una matriz se usa una tubería y para recibirla otra.

Podemos decir que en el sistema operativo Windows se muestra casi la misma versatilidad en tuberías y memoria compartida que en Linux, pero este eleva su complejidad de uso.

5. Conclusiones

La memoria compartida y las tuberías son de gran ayuda para compartir información entre distintos procesos, pero cada uno de estos mecanismos aporta diferentes características, debido a su forma de operar, por lo tanto, se debe conocer adecuadamente su sintaxis y funcionamiento para poder aprovechar sus funcionalidades. Las tuberías pueden enviar información a procesos hijos, pero estas son únicamente unidireccionales, por lo que es necesario crear una para cada comunicación entre ellos, es decir, de ida como de vuelta. Por otro lado, la memoria compartida proporciona más posibilidades, ya que su funcionamiento es similar a la memoria dinámica, se asigna a un apuntador para acceder a ella mediante la llave, la cual sirve para localizarla y acceder, teniendo menos restricciones que las tuberías.