Código memoria compartida Linux

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

#include <math.h>

#include <unistd.h>

#include <sys/wait.h>

#include "Funciones2.h"

#define TAM\_MEM 200 /\*Tamaño de la memoria compartida en bytes\*/

#define TAM 10

int main (void){

pid\_t pid;

char \*argv[2];

argv[0] = "/home/alexis/Documentos/programas\_sistemas\_operativos/practica6/Hijo";

argv[1]=NULL;

int j,i,m1[TAM][TAM];

int m2[TAM][TAM];

int shm\_id,shm\_id2;

int \*shm,\*shm2,\*apT;

float inv[TAM][TAM];

key\_t llave=123,llave2=456;

////////////////////////////////////reservar espacios de memoria para el hijo////////////////////

if((shm\_id=shmget(llave,TAM\_MEM,IPC\_CREAT | 0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm=shmat(shm\_id,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

if((shm\_id2=shmget(llave2,TAM\_MEM,IPC\_CREAT | 0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm2=shmat(shm\_id2,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

generaMatriz(m1,m2);

////////////////////////envio de la matriz uno al hijo//////////////////////////////////////

apT=shm+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

\*apT++=m1[i][j];

}

}

printf("\n matriz uno enviada\n");

imprimeMatriz(m1);

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

apT=shm2+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

\*apT++=m2[i][j];

}

}

printf("\n matriz 2 enviada\n");

imprimeMatriz(m2);

if((pid = fork()) == 0)

execv(argv[0],argv);

wait(0);

while (\*shm!=-1)

sleep(1);

apT=shm+1; //copiar el resultado de la multiplicacion

//sobreescribiendo en m1

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

m1[i][j]=\*apT++;

}

}

printf("\n matriz multiplicada recibida desde el hijo\n");

imprimeMatriz(m1);

while (\*shm2!=-1)

sleep(1);

apT=shm2+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

m2[i][j]=\*apT++;

}

}

printf("\n matriz sumada recibida desde el nieto \n");

imprimeMatriz(m2);

double determinante1;

double determinante2;

double inv1[10][10]; /\* Matrices auxiliares para el calculo de la inversa \*/

double inv2[10][10];

for(i=0 ; i<10 ; i++ )

{

for(j=0 ; j<10 ; j++ )

{

inv1[i][j]=m1[i][j];

inv2[i][j]=m2[i][j];

}

}

determinante1=Determinante(inv1, 10);

if (determinante1!=0)

{

matrizCofactores(inv1, 10, determinante1);

}

else

{

printf("La matriz 1 no tiene inversa\n");

}

determinante2=Determinante (inv2, 10);

if (determinante2!=0)

{

matrizCofactores2(inv2, 10, determinante2);

}

else

{

printf("La matriz 2 no tiene inversa\n");

}

printf("He terminado, los resultados se han guardado en su archivo correspondiente.\n");

system("pause");

shmdt ((char \*)shm);

shmdt ((char \*)shm2);

shmctl(shm\_id, IPC\_RMID, NULL);

shmctl(shm\_id2, IPC\_RMID, NULL);

exit(0);

}

Código del hijo memoria compartida Linux

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

#include <math.h>

#include <unistd.h>

#include <sys/wait.h>

#include "Funciones2.h"

#define TAM\_MEM 200 /\*Tamaño de la memoria compartida en bytes\*/

#define TAM 10

int main (int argc , char \* argv[]){

pid\_t pid;

char \*argv2[2];

argv2[0] = "/home/alexis/Documentos/programas\_sistemas\_operativos/practica6/Nieto";

argv2[1]=NULL;

int j,i,m1[TAM][TAM];

int m2[TAM][TAM],mult[TAM][TAM],m3[TAM][TAM],m4[TAM][TAM];

int shm\_id,shm\_id2,shm\_id3,shm\_id4;

int \*shm,\*shm2,\*shm3,\*shm4,\*apT;

key\_t llave=123,llave2=456,llave3=789,llave4=666;

////////////////////acceder a los espacios de memoria del pade /////////////////////////

if((shm\_id=shmget(llave,TAM\_MEM,0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm=shmat(shm\_id,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

if((shm\_id2=shmget(llave2,TAM\_MEM,0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm2=shmat(shm\_id2,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

//////////////////////////////////////////////////////////////////////////////////////////////////

///////////////////////////////reservar espacios de memoria para el nieto////////////////////

if((shm\_id3=shmget(llave3,TAM\_MEM,IPC\_CREAT | 0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm3=shmat(shm\_id3,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

if((shm\_id4=shmget(llave4,TAM\_MEM,IPC\_CREAT | 0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm4=shmat(shm\_id4,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

////////////////////////////////////////////////////////////////////////////////////////////////////

generaMatriz(m3,m4);

apT=shm+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

m1[i][j]=\*apT;

apT=apT+1;

}

}

printf("\nmatriz1 recibida en el hijo\n ");

imprimeMatriz(m1);

printf("\nse multiplicara con la matriz 2 \n ");

////////////////////////////////////////

apT=shm2+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

m2[i][j]=\*apT;

apT=apT+1;

}

}

imprimeMatriz(m2);

multiplicaMatrices(m1,m2,mult);

apT=shm+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

\*apT++=mult[i][j];

}

}

\*shm=-1;

apT=shm3+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

\*apT++=m3[i][j];

}

}

printf("\n matriz 3 enviada al nieto\n");

imprimeMatriz(m3);

apT=shm4;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

\*apT++=m4[i][j];

}

}

printf("\n matriz 4 enviada al nieto\n");

imprimeMatriz(m4);

if((pid = fork()) == 0)

execv(argv2[0],argv2);

wait(0);

while (\*shm3!=-1)//espera señal del nieto para

sleep(1);

\*shm2=-1;//enviar señal de continuar al padre

shmdt ((char \*)shm);

shmdt ((char \*)shm2);

shmdt ((char \*)shm3);

shmdt ((char \*)shm4);

shmctl(shm\_id, IPC\_RMID, NULL);

shmctl(shm\_id2, IPC\_RMID, NULL);

shmctl(shm\_id3, IPC\_RMID, NULL);

shmctl(shm\_id4, IPC\_RMID, NULL);

exit(0);

}

Código nieto memoria Compartida Linux

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <stdbool.h>

#include <math.h>

#include <unistd.h>

#include <sys/wait.h>

#include "Funciones2.h"

#define TAM\_MEM 200 /\*Tamaño de la memoria compartida en bytes\*/

#define TAM 10

int main (int argc , char \* argv2[]){

int j,i,m3[TAM][TAM];

int m4[TAM][TAM],sum[TAM][TAM];

int shm\_id2,shm\_id3,shm\_id4;

key\_t llave2=456,llave3=789,llave4=666;

int \*shm2,\*shm3,\*shm4,\*apT;

////////////////////acceder a los espacios de memoria del hijo del padre /////////////////////////

if((shm\_id3=shmget(llave3,TAM\_MEM,0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm3=shmat(shm\_id3,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

if((shm\_id2=shmget(llave2,TAM\_MEM,0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm2=shmat(shm\_id2,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

if((shm\_id4=shmget(llave4,TAM\_MEM,0666)) < 0)

{

perror("1-Error al obtener memoria compartida: shmget");

exit(-1);

}

if((shm4=shmat(shm\_id4,NULL,0))==(int\*)-1)

{

perror("Error al enlazar la memoria compartida: shmat");

exit(-1);

}

///////////////////////////////////////////////////////////////////////////////

apT=shm3+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

m3[i][j]=\*apT;

apT=apT+1;

}

}

printf("\n matriz 3 recivida del hijo del padre:\n ");

imprimeMatriz(m3);

printf("\nse sumara con matriz 4 :\n ");

////////////////////////////////////////

apT=shm4;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

m4[i][j]=\*apT;

apT=apT+1;

}

}

imprimeMatriz(m4);

sumaMatrices(m3,m4,sum);

apT=shm2+1;

for ( i=0; i<TAM; i++)

{

for ( j=0; j<TAM; j++)

{

\*apT++=sum[i][j];

}

}

\*shm3=-1;

// system("pause");

shmdt ((char \*)shm2);

shmdt ((char \*)shm3);

shmdt ((char \*)shm4);

shmctl(shm\_id2, IPC\_RMID, NULL);

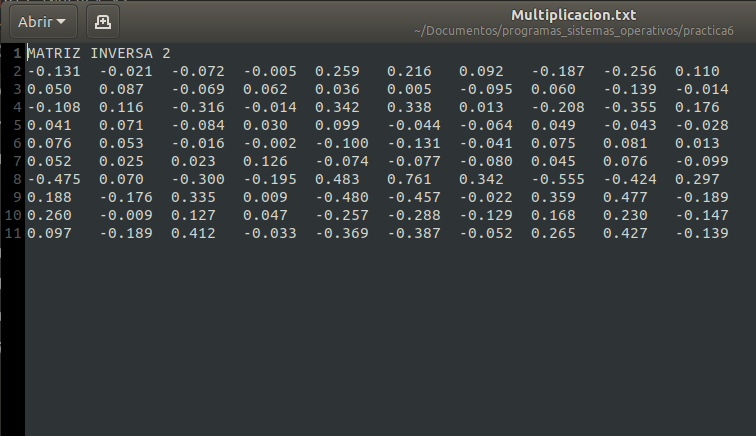
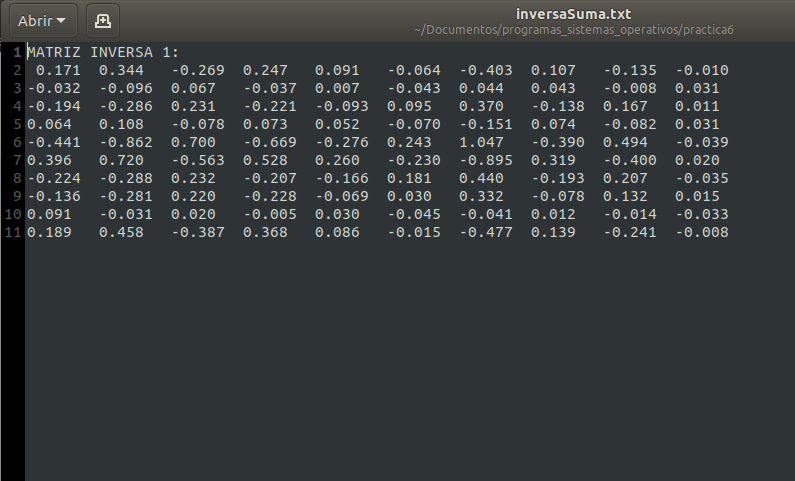
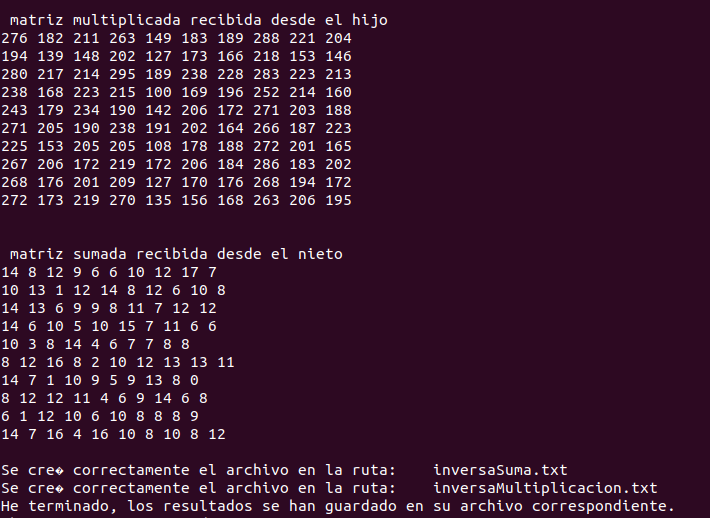
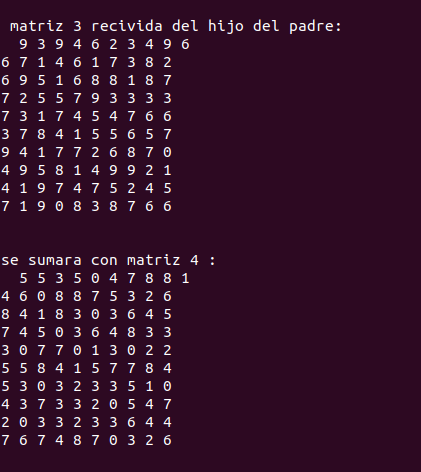
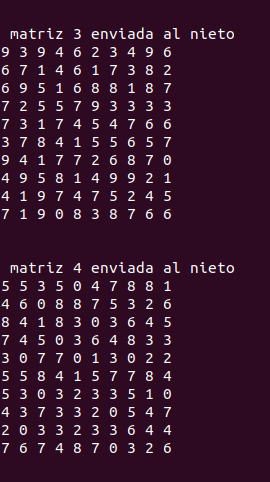
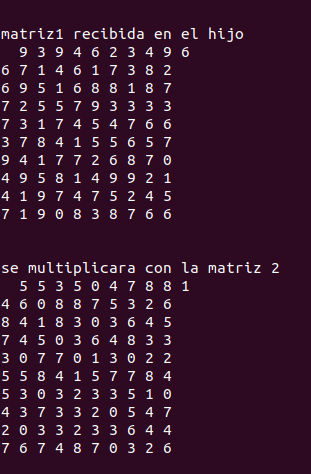
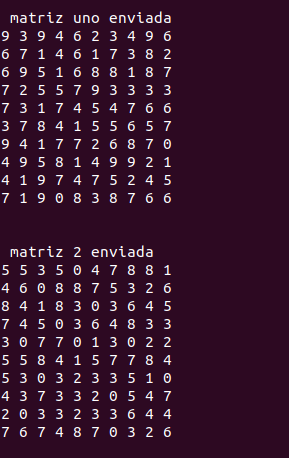
shmctl(shm\_id3, IPC\_RMID, NULL);

shmctl(shm\_id4, IPC\_RMID, NULL);

exit(0);

}

Capturas de pantalla memoria compartida Linux



Observaciones memoria compartida Linux

* Para este programa al igual que en Windows se reservaron cuatro espacios (uno para cada matriz para facilitar el uso de las matrices por separado y no en una estructura) de memoria para que trabajaran los tres programas. El padre crea dos espacios de memoria , que inicialmente servirán para enviar las primeras dos matrices al espacio de memoria compartido con el hijo y posteriormente en estos dos espacios del padre se guardaran los resultados enviados desde el hijo y su nieto. El hijo reserva dos espacios más para enviar otras dos matrices al nieto y accede a los dos iniciales del padre para realizar operaciones. La diferencia con respecto a su versión de Windows son as llamadas al sistema para reservar y acceder a los espacios de memoria, ademas del punto final cuando el padre tiene que escribir en un archivo los resultados, para esto se modifico el archivo de funciones.h pues cambian también las llamadas al sistema que escriben archivos. El único inconveniente presentado en esta practica fue referente al tema de las banderas de “sincronización” pues para salir de los ciclos while de espera se imprimia el -1 en la primera posición de la matriz recibida usado como indicador de continuar, para ello se guardo este en la primera posición de la memoria compartida y a partir de la segunda se guardo como vector unidimensional las matrices en sus correspondientes espacios reservados.

Conclusiones

Lara Delgado Edgar Alexis

En esta penúltima practica del curso se revisó la comunicación entre procesos en ambos sistemas operativos, para lograrlo se usaron tuberías y memoria compartida. Se implementaron múltiples tuberías como primer método para la resolución del problema planteado pues estas comunican únicamente en una sola dirección y también solo pueden comunicar dos procesos a la vez, se usaron los conocimientos de las dos practicas anteriores ademas de la librería propia funciones.h para hacer las operaciones de los datos compartidos entre procesos.

Como segundo método de resolución al problema se hizo uso de memoria compartida donde con ayuda de las regiones compartidas los múltiples procesos (padre,hijo,nieto) accedían a los datos almacenados para poder comunicarse, a diferencia de las tuberias estos canales se usaron bidireccional mente, pues se enviaron matrices desde el padre en una región y en esta misma se sobrescribio el resultado recibido desde el hijo y el nieto