1. #include <unistd.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdlib.h>

#include <stdio.h>

**/\*Memoria Compartida emparentada**

**\*/**

int main(){

int child,sdId;

size\_t size=11;

key\_t key=1 ;

char \*msj;

sdId=shmget(key,size,0666|IPC\_CREAT); //pedir el espacio en memoria

if (sdId==1){

perror("error shmget:");

}

msj=shmat(sdId,0,0);

/\*if(msj==-1){

perror("error shmmat:");

}

\*/

child = fork(); /\*Creacion subproceso\*/

if (child == -1){

error("Could not fork, QUIT!");

}

if(child == 0){ /\*Child process \*/

sprintf(msj,"Hola mundo");

printf("%s","child");

}else { /\* This is the parent Process \*/

sleep(2);

printf("%s",msj);

printf("%s","father");

}

//shmdt(msj);

}

**/\*Programa para compartir memoria (escribe el abecedario en ese espacio)\*/**

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#define SHMSZ 27

int main()

{

char c;

int shmid;

key\_t key;

char \*shm, \*s;

/\*

\* Nuestro segmento de Memoria compartida sera

\* "5678".

\*/

key = 5678;

/\*

\* Creacion del segmento -1 signfica error

\*/

if ((shmid = shmget(key, SHMSZ, IPC\_CREAT | 0666)) < 0) {

perror("shmget");

exit(1); //fin ejecucion

}

/\*

\* Ahora guardamos el id de nuestro attach de memoryshared

\*/

if ((shm = shmat(shmid, NULL, 0)) == (char \*) -1) {

perror("shmat");

exit(1);

}

/\*

\*guardaremos algo el la memory shared para que el otro

\* proceso lea

\*/

s = shm; /\*Lo copiamos para no perder el original \*/

for (c = 'a'; c <= 'z'; c++)

\*s++ = c;

\*s = '\0';

/\*

\*Nuestra bandera sera el cambio de los caracteres por \*

\*por tanto esperamos a que el otro proceso logre

\*odificar este

\*/

while (\*shm != '\*')

sleep(1);

exit(0);

}

**/\***

**\* shm-client - lee un espacio de memoria compartido conocido por el KEY**

**\* 5678 y lee lo que esta alli hasta encontrar el caracter '\0'**

**\*/**

#include <sys/types.h>

#include <sys/ipc.h>

#include <sys/shm.h>

#include <stdio.h>

#include <stdlib.h>

#define SHMSZ 27

main()

{

int shmid;

key\_t key;

char \*shm, \*s;

/\*

\*Nombre del segmento que estamos usando

\*/

key = 5678;

/\*

\* Solicitamos al SO que nos de el ID del espacio en memoria

\*con KEY =5678 y del mismo tamaño.

\*/

if ((shmid = shmget(key, SHMSZ, 0666)) < 0) {

perror("shmget");

exit(1);

}

/\*

\* Hacemos el attach del espacio de memoria y validamos que sea

\*correcto .

\*/

if ((shm = shmat(shmid, NULL, 0)) == (char \*) -1) {

perror("shmat");

exit(1);

}

/\*

\* Leemos el espacio en memoria

\*/

for (s = shm; \*s != '\0'; s++)

putchar(\*s);

putchar('\n');

/\*

\* Levantamos la bandera para indicar que la lectura finalizo

\*/

\*shm = '\*';

exit(0);

}

#include<stdio.h>

#include<unistd.h>

#include<errno.h>

#include<stdlib.h>

#include<pthread.h>

#define ITER 1e09

**/\*Calculo PI con comunicacion mediante archivo\*/**

int cal(double\*pi,long int init ,int end){ //funcionCalculaMitadPI

\*pi=0.0;

for (i=init;i<end;){

\*pi=\*pi-(double)(4.0)/(double)(i+=2);

\*pi=\*pi+(double)(4.0)/(double)(i+=2); // pow (-1.i) -> hace mas lento codigo

}

return 0;

}

int main(){

FILE \*ap; //Creacion Pointer Archivo

pid\_t pid;

int r; // retorno de FILE

double a,b;

pid=fork(); //Creación Hijo

if(pid==-1){

perror("No fue posible crear el proceso Hijo:");

}

if(pid==0){

cal(&b,(ITER/2)+1,ITER);

ap=fopen("data.dat","w+"); //Crea un archivo vacio para leer y escribir

if (ap == NULL){ //validacion apertura archivo

perror("No se puede crear o abrir el archivo porque:");

exit(-1);

}

r=fwrite((void\*)&b,sizeof(double),1,ap);//escribe en el archivo el calculo realizado

r=fclose(ap); //Cierra el archivo

exit(0); //terminacion programa

}else{

cal(&a,1,(ITER/2));

do{

ap=fopen("data.dat","w+");

}while(ap==NULL); //valida hasta que el archivo exista

do{

r=fread(&b,sizeof(double),1,ap);

}while(r!=1); //hasta q logre leer todos los bytes del dato

r=fclose(ap); //cierra archivo

if ( r != 0){

perror("No se puede cerrar el archivo");

}

a=a+b; // suma los 2 datos

printf("%lf",a);

}

}

#include<stdio.h>

#include<unistd.h>

#include<errno.h>

#include<math.h>

#include<stdlib.h>

#include<pthread.h>

**/\*LECTURA Y ESCRITURA DE ARCHIVO emparentado\*/**

int main(){

FILE \*ap; // Archivo a escribir

int pid = fork(); //creacion\_Hijo

int r; //controlador error

if(pid == -1){

perror("Error en fork");

exit(0);

}

ap =fopen("texto.dat" , "w+");

if (ap== NULL){

perror("No se puede crear o abrir el archivo porque:");

exit(0);

}

if(pid== 0){ //proceso hijo

printf("Hijo");

fwrite("Hijo", sizeof(char)\*4,4, ap); //escribir el archivo

}else{ //proceso padre

printf("Padre");

fwrite("Padre", sizeof(char)\*5,4, ap);

}

r = fclose(ap); // Ordeno cerrar el archivo para no escribir mas

if ( r != 0){

perror("No se puede cerrar el archivo");

}

}

**/\* lector\_fifo.c - Tuberia con nombre usando mkfifo \*/**

#include <stdlib.h>

#include <fcntl.h>

//#include <sys/mode.h>

int linea ();

/\*mkfifo escritorr\*/

int main ()

{

int descr;

char cadena[100];

unlink ("tuberia"); /\*Desligamos la tuberia nombrada \*/

mkfifo ("tuberia", 0); /\*Creamos una tuberia sin permisos\*/

chmod ("tuberia", 460); /\*Cambiamos los permisos en "Tuberia"\*/

descr = open ("tuberia", O\_RDONLY); /\*Abrimos la tuberia\*/

while (linea (descr, cadena))

printf ("%s\n", cadena);

close (descr);

pritnf ("Fin del lector.\n");

}

int linea (int df, char \*cad)

{

int n;

do

{

n = read (df, cad, 1); /\*Leemos de a 1 byte y guardamos en cad\*/

}

while (n > 0 && \*cad++ != NULL);

return (n > 0);

}

#include <stdlib.h>

#include <fcntl.h>

#include <string.h>

//#include <sys/mode.h>

**/\*Mkfifo lector tuberia NO emparentada\*/**

int main ()

{

int descr, longmens, i;

char \* mensaje;

mensaje="HolaMundo";

//printf ("Un saludo desde el proceso %d", getpid ());

longmens = strlen (mensaje) + 1;

do

{ /\* intentar la conexion \*/

descr = open ("tuberia", O\_WRONLY);

if (descr == -1) sleep (1);

}

while (descr == -1);

for (i=0; i<3; i++)

{

write (descr, mensaje, longmens);

sleep (3);

}

close (descr);

//printf("Fin del escritor %d\n", getpid ());

}

#include <pthread.h>

#include<stdio.h>

#include<stdlib.h>

#include <time.h>

**/\*Hilos pthread COREHOT\*/**

#define NUMTHREADS 8

void \* funcion(void \* arg ){

int i=0;

srand(time(NULL));

int r = rand();

for(i=0;i<10000000;i++){

char \* name="holamundocfhavgjskdjshfklsagljjdsgnkjds";

unsigned long nHash = 0;

int i;

for ( i= 1000 - 1; i >= 0; --i){

nHash = r % 1000000007;

r = rand();

}

printf("hola mundo %i \n" ,r % 1000);

printf("hola mundo %c \n " ,\*(int\*)arg);

}

}

int main(){

pthread\_t hilo;

int arg;

arg=0;

int i,r;

for(i=0;i<NUMTHREADS;i++){

r=pthread\_create(&hilo,NULL,funcion,(void \*)&arg);

if(r<0){

perror("error\_pthread\_create");

}

}

for(i=0;i<NUMTHREADS;i++){

r=pthread\_join(hilo,NULL);

if(r!=0){

perror("error\_pthread\_join");

}

}

}

#include <sys/types.h>

#include <fcntl.h>

#include <unistd.h>

#include <stdio.h>

#include<stdlib.h>

#include<string.h>

**/\*Comucacion de procesos EMPARENTADOS mediante tuberias PIPE**

**\*En este caso calculamos PI**

**\*/**

#define ITER 1e09

#define SIZE 40

int calc(double\*pi,long int init ,int end){ //funcionCalculaMitadPI

if(init == 1){

\*pi=4.0;

}else{

\*pi=0.0;

}

int i;

for (i=init;i<end;){

\*pi=\*pi-(double)(4.0)/(double)(i+=2);

\*pi=\*pi+(double)(4.0)/(double)(i+=2); // pow (-1.i) -> hace mas lento codigo

}

return 0;

}

int main( int argc, char \*\*argv )

{

pid\_t pid;

int p[2], readbytes;

double \*pi;

double num ;

pipe( p );

pi=&num;

pid=fork(); /\*Creacion Subproceso hijo \*/

if ( (pid) == 0 )

{ /\* hijo\*/

close( p[0] ); /\* cerramos el lado de lectura del pipe \*/

calc(pi,1,(ITER/2));

write( p[1] , pi, sizeof(pi) );/\*Escribimos en el lado de escritura\*/

close( p[1] ); /\*cerramos lado escritura\*/

}

else

{ /\* padre\*/

double a;

close( p[1] ); /\*cerramos lado escritura\*/

calc(&a,(ITER/2)+1,ITER);

read(p[0],pi,sizeof(pi));/\*Leemos \*/

printf("%.\*f",10 ,\*pi+a); /\*Mostramos el calculado y el leido \*/

close( p[0] );//cerramos lado lectura

}

exit( 0 );

}

#include <sys/types.h>

#include <sys/socket.h>

#include<unistd.h>

#include<errno.h>

#include<stdio.h>

#include <strings.h>

#include <netinet/in.h>

#define PORT 3535

#define BACKLOG 0

**/\*Configuración Socket MODO SERVIDOR\*/**

int main(){

int serverfd,r,clientfd;

struct sockaddr\_in server,client;

socklen\_t size =sizeof(struct sockaddr\_in);

socklen\_t size\_cli;

char buffer[32];

serverfd=socket(AF\_INET,SOCK\_STREAM,0);/\*creación server\*/

if(serverfd == -1){

perror("Error Socket");

}

server.sin\_family =AF\_INET;

server.sin\_port=htons(PORT);

server.sin\_addr.s\_addr=INADDR\_ANY;

bzero(server.sin\_zero,8);//rellena de zeros

r=bind(serverfd,(struct sockaddr \*)&server,size);//bautizo socket

if(r<0){

perror("error bind"); }

r=listen(serverfd,BACKLOG); //configguracion de escucha

if(r<0){

perror("error listen ");

}

clientfd=accept(serverfd,(struct sockaddr \*)&client,(socklen\_t\*)& size\_cli);//crear la creacion para recibir msj

if(clientfd<0){perror("err");}

r=recv(clientfd,buffer,32,0);

if(r<=0){

perror("error");

}

printf("%s",buffer);

close(clientfd);

close(serverfd);

}

#include <stdio.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <strings.h>

#include <netinet/in.h>

#define PORT 3535

**/\*Socket MODO CLIENTE \*/**

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

//argv[0]Nombre executable,[1]primer parametro VA EL PUERTO (BACKLOG),[2]segundo parametro separado por espacio\*/

int clientfd,r;

clientfd=socket(AF\_INET,SOCK\_STREAM,0);

/\* \*AF\_INET especifica coneccioón mediante ip\_v4

\*SOCK\_STREAM especifica tipo de tcp

\*/

struct sockaddr\_in client;

socklen\_t size =sizeof(struct sockaddr\_in );

if(clientfd==0){

perror("error socket");

}

client.sin\_family=AF\_INET;

client.sin\_port=htons(PORT); /\*htons soluciona problema envianismo\*/

client.sin\_addr.s\_addr=inet\_addr(argv[1]); //pasamos la direccion por parametro en el ./

bzero(client.sin\_zero,8);

r=connect(clientfd,(struct sockaddr \*)&client,size);

if(r==-1){

perror ("ERROR in conect");/\*error al conectarse al servidor \*/

}

/\*\*/

r=send(clientfd,"hola mundo",10,0);

if(r!=10){ //no se envio el paquete completo

;

}

close(clientfd);

}