Sistemas Operativos

Formulario de auto-evaluación

Modulo 2. Sesión 4. Comunicación entre procesos utilizando cauces

Nombre y apellidos:

Cristina María Garrido López

a) Cuestionario de actitud frente al trabajo.

El tiempo que he dedicado a la preparación de la sesión antes de asistir al laboratorio ha sido de 30 minutos.

1. He resuelto todas las dudas que tenía antes de iniciar la sesión de prácticas: sí. En caso de haber contestado “no”, indica los motivos por los que no las has resuelto:

2. Tengo que trabajar algo más los conceptos sobre:

Dup y dup2

3. Comentarios y sugerencias:

b) Cuestionario de conocimientos adquiridos.

Mi solución al **ejercicio 1** ha sido:

Primero hay que ejecutar el consumidor FIFO de manera que se cree nuestro archivo ComunicacionFIFO, a continuación, ejecutamos el productor FIFO pasándole un argumento. Cuando volvamos a ejecutar el consumidor lo mostrará por pantalla, y así sucesivamente hasta que reciba la cadena “fin”.

Queda reflejado en el sistema que estamos utilizando un cauce FIFO porque el archivo ComunicacionFIFO sigue ahí.

Mi solución a la **ejercicio 2** ha sido:

El ejercicio 2 crea un cauce sin nombre por medio de la orden pipe pasándole fd, que crea un descriptor de lectura y otro de escritura. Crea también un proceso hijo con fork que va a encargarse de enviar el mensaje a partir del cauce de escritura, cerrando el descriptor de lectura. El proceso padre cierra el descriptor de escritura y lee el de escritura, sacando por pantalla el número de bytes que tiene y el propio mensaje que envió el hijo.

Mi solución a la **ejercicio 4** ha sido:

La principal diferencia en el código es la utilización de dup2 en tarea8 y dup en tarea7, además de close, ya que en dup es necesario haber cerrado el descriptor por defecto anteriormente. No se puede apreciar ninguna diferencia en su ejecución. La llamada a dup2 primero cierra el descriptor antiguo y seguidamente lo duplica. Así el proceso hijo va a ejecutar la orden ls en el descriptor de escritura y el padre ejecuta sort para este mensaje que envió el hijo.

Mi solución a la **ejercicio 5** ha sido:

Esclavo.c

```
#include<sys/types.h>

#include<fcntl.h>

#include<unistd.h>

#include<stdio.h>

#include<stdlib.h>

#include<errno.h>

#include <string.h>

#include <math.h>




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

```
	int inicio, fin,i,j;
	double lim;
	int primo;
	char imprime[10];

	if(argc!=3){
		printf("uso: ./esclavo <inicio> <fin>");
		exit(0);
	}

	inicio = atoi(argv[1]);
	fin = atoi(argv[2]);

	for(i=inicio; i<fin; i++){
		lim=sqrt(i);
		primo=1;
		for (j = 2; j <= lim && primo; j++)
			if (i % j == 0)
				primo = 0;
		if(primo){
			sprintf(imprime,"%d ",i);
			write(STDOUT_FILENO,imprime,strlen(imprime)+1);

		}

	}
	exit(0);
}
```

Maestro.c

```
#include<sys/types.h>
#include<fcntl.h>
#include<unistd.h>
#include<stdio.h>
#include<stdlib.h>
#include<errno.h>
#include<string.h>
#include<math.h>

int main(int argc, char* argv[]){
        pid_t esclavo1, esclavo2;
        int intervalo[2],intesc1[2],intesc2[2],buf1,buf2;
        int fd1[2];
        int fd2[2], bl1,bl2;
        char ini[10],fin[10];

        if(argc != 3){
                printf("uso: ./maestro <inicio> <fin>\n");
                exit(0);
        }
        if(setvbuf(stdout,NULL,_IONBF,0)) {
                perror("\nError en setvbuf");
        }
        intervalo[0]=atoi(argv[1]);
        intervalo[1]=atoi(argv[2]);
        intesc1[0]=intervalo[0];
        intesc1[1]=(intervalo[0]+intervalo[1])/2;
        intesc2[0]=(intervalo[0]+intervalo[1])/2+1;
        intesc2[1]=intervalo[1];
```

```
pipe(fd1);
pipe(fd2);

printf("Los números en el intervalo [%d,%d] son:\n",intesc1[0],intesc1[1]);

if ((esclavo1=fork())<0) {
        perror("\Error en fork");
        exit(0);
}

if(esclavo1==0){
        close(fd1[0]);//cierro lectura
        //close(1);
        dup(fd1[1]);
        sprintf(ini,"%d",intesc1[0]);
        sprintf(fin,"%d",intesc1[1]);
        if(execl("./esclavo", "esclavo", ini, fin, NULL) < 0) {
                perror("\nError en el execl");
                exit(-1);
        }
}else{
        close(fd1[1]);
        while((read(fd1[0],&buf1,sizeof(int)))>0)
                printf("%d",buf1);
        close(fd1[0]);

        printf(" ");

        printf("Los números en el intervalo [%d,%d] son:\n",intesc2[0],intesc2[1]);
```

```
		if ((esclavo2=fork())<0) {
			perror("\Error en fork");
			exit(0);
		}
		if(esclavo2==0){
			//close(STDOUT_FILENO);
			close(fd2[0]);
			dup(fd2[1]);
			sprintf(ini,"%d",intesc2[0]);
			sprintf(fin,"%d",intesc2[1]);
			execl("./esclavo","esclavo",ini,fin,NULL);
		}else{
			close(fd2[1]);
			while((read(fd2[0],&buf2,sizeof(int)))>0)
				printf("%d",buf2);

			printf("\n");
			close(fd2[0]);
		}
	}
	printf(" ");
	exit(0);
}
```