Sistemas Operativos

Formulario de auto-evaluación

Modulo 2. Sesión 4. Comunicación entre procesos utilizando cauces

Nombre y apellidos:

David Sánchez Jiménez

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 …… minutos.

1. He resuelto todas las dudas que tenía antes de iniciar la sesión de prácticas: …… (si/no). 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:

3. Comentarios y sugerencias:

b) Cuestionario de conocimientos adquiridos.

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

La manera correcta de ejecutar el productor/consumidor es la siguiente:
$ ./productorFIFO mensaje | ./consumidorFIFO

El programa consumidorFIFO crea el archivo ComunicacionFIFO. El productorFIFO accede al archivo ComunicacionFIFO una vez creado donde almacena la cadena mensaje. Por ultimo el consumidorFIFO lee el mensaje escrito dicho archivo.

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

El programa tarea6 crea un cauce con la orden pipe y le pasa como parámetro el vector fd.
A continuación crea un hijo con la orden fork, comprueba que el PID del hijo sea 0, cierra su descriptor de lectura y usa el descriptor de escritura para enviar el mensaje mediante un write.
El padre cierra el descriptor de escritura y lee los datos del cauce enviados por el hijo con el descriptor de lectura usando read, imprimiendo el la cadena recibida numero de bytes de esta.

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

Este programa es similar al anterior pero usando *dup2* el cual nos permite indicar explícitamente el descriptor que vamos a usar en lugar de coger el primer descriptor que estuviera libre.
Después de realizar el *fork* se redirige la salida estándar al file descriptor que queremos, ejecuta *ls* en el hijo , el padre lo recibe desde la entrada estándar y se lo pasa a *sort*.

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

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

int esPrimo(int n) {
  int raiz = sqrt(n);
  int es_primo = 1;

  for (int i = 2; i <= raiz && es_primo; i++) {
    if (n % i == 0) {
      es_primo = 0;
    }
  }

  return es_primo;
}

int main(int argc, char const *argv[]) {
  if (argc != 3) {
    printf("Modo de uso: %s <inicio> <fin>\n\n", argv[0]);
    exit(-1);
  }

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

  for (int i = inicio; i < fin; i++) {
    if (esPrimo(i)) {
      printf("El numero %d es primo.\n", i);
    }
  }

  return 0;
}
```

```
// Maestro
#include <errno.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#define ARCHIVO_FIFO "comunicacion";
int main(int argc, char *argv[]) {
  int inicio, fin, mitad, numBytes;
  int fd1[2];
  int fd2[2];
  char buffer[80];
  char params[2];
  if (argc != 3) {
    printf("Modo de uso: %s <inicio> <fin>\n\n", argv[0]);
    exit(1);
  }
  inicio = atoi(argv[1]);
  fin = atoi(argv[2]);
  mitad = inicio + ((fin - inicio) / 2);
  pid_t PID;
  pipe(fd1);
  pipe(fd2);
  if ((PID = fork()) < 0) {
    perror("Error al hacer fork");
    exit(1);
  }
  if (PID == 0) {
    close(fd1[0]);
    params[0] = inicio;
    params[1] = mitad;
    printf("paso 1\n");
    dup2(fd1[1], STDOUT_FILENO);
    execlp("esclavo", "esclavo", params, NULL);
  } else {
    if ((PID = fork()) < 0) {
      perror("Error al hacer fork");
      exit(1);
    }
    if (PID == 0) {
      close(fd2[0]);
      params[0] = mitad + 1;
      params[1] = fin;
      printf("paso 2\n");
      dup2(fd2[1], STDOUT_FILENO);
      execlp("esclavo", "esclavo", params, NULL);
      exit(0);
    } else {
      close(fd1[1]);
      numBytes = read(fd1[0], buffer, sizeof(buffer));
      dup2(fd1[0], STDIN_FILENO);

      printf("paso 3\n");
```

```
    while ((numBytes = read(fd1[0], &buffer, 4)) > 0) {
      printf("Valor: %d", 1);
    }
   }
  }

  exit(0);
}
```