Escuela Superior de Cómputo
Instituto Politécnico Nacional

Sistemas Operativos

Reporte Práctica 5

Alumno:

González Barrios Alan Ernesto

2CM7
April 9, 2019

April 9, 2019

1 Índice

2 Explicación de la teoría detras de la práctica

En informática, una tubería (pipeline) consiste en una cadena de procesos conectados de forma tal que la salida de cada elemento de la cadena es la entrada del próximo. Permiten la comunicación y sincronización entre procesos. Es común el uso de búfer de datos entre elementos consecutivos.

Las tuberías (pipes) están implementadas en forma muy eficiente en los sistemas operativos multitarea, iniciando todos los procesos al mismo tiempo, y atendiendo automáticamente los requerimientos de lectura de datos para cada proceso cuando los datos son escritos por el proceso anterior. De esta manera el planificador de corto plazo va a dar el uso de la CPU a cada proceso a medida que pueda ejecutarse minimizando los tiempos muertos.

3 Explicación de la lógica de programación y las sálidas obtenidas

Para realizar la practica se tenian que crean numers aletorios estos siendo un total de n veces, que n sera la entrada del programa y la salida de este sera la suma total que sera devuelta de los hijos del padre proceso original, para que estos hijos puedan hacer sus calculos el padre en base a los numeros aleatorios que genero va a verificar si el numero que se genero es impar o par y dependiendo de lo que sean el proceso padre determinara a que proceso hijo mandarlo.

Para que todo esto funcione se tuvieron que crear 4 pipes uno que es la comunicacion hijo1 a padre, otro padre a hijo1, otro que es de hijo2 a padre y finalmente de padre a hijo2, y en cada uno de estos en su seccion correspondiente de codigo se les cerro los canales que no se ocuparan de cada pipe, puesto que unos pipes solo escucharan y otros solo escribiran, y para que funcione indeterminadamente se tuvo que poner un ciclo infinito en los hijos para que estos funcionen de escuchas al padre en todo momento y para disntinguir cuando es que tienen que mandar la suma total, cuando se generaron los randoms se aseguro que se genero todo menos el 0, porque se tiene la intención de que el 0 sirva como el identificador para que los hijos sepan cuando mandar la suma y cerrarse.

Una vez recibido el 0, solo el padre los lee de los pipes correspondientes y lo imprime en pantalla.

4 Listado del código, sálidas a consola y pantallas de resultado

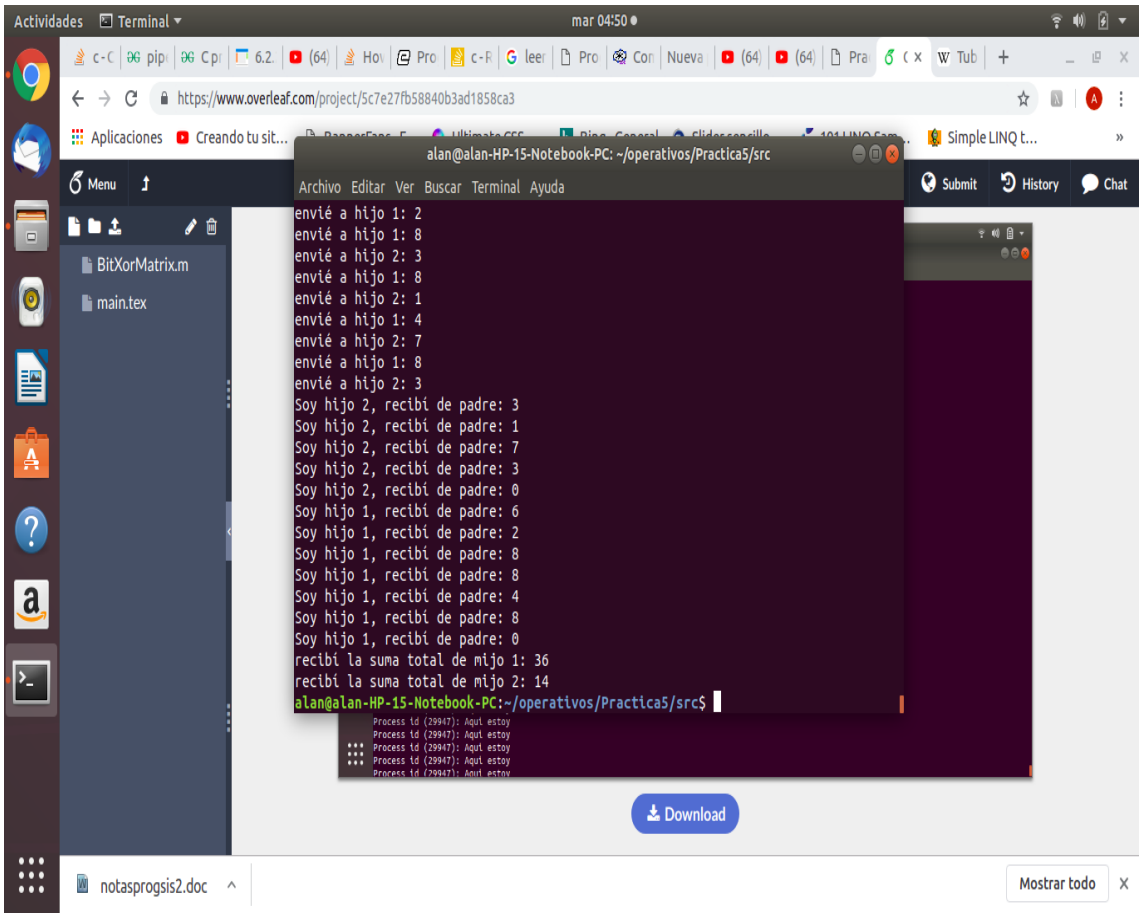
El siguiente codigo corresponde al programa desarrollado en C, el cual tiene como funcion imprimir de manera infinita una cadena con su id correspondiente.

```
#include <unistd.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

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

  int h1p[2], ph1[2], h2p[2], ph2[2], n, i, aux=0, a, b;
  int leidos=0;
  printf("Ingresa un n mero mayor a 0\n");
  scanf("%d",&n);
  while(n==0){
    printf("Ingresa otro valor mayor a 0");
//! recibe n
    scanf("%d",&n);
  }
//! se crean los pipes
  pipe(h1p);
  pipe(ph1);
  pipe(h2p);
  pipe(ph2);
//! se crea el hijo 1
  if(fork()==0){
//! se cierran los canales
    close(h1p[0]);
    close(ph1[1]);
//! en un ciclo infinito escucha lo que le manda padre
    while(1){
      read(ph1[0],&a,sizeof(int));
      printf("Soy hijo 1, recib  de padre: %d\n",a);
//! incrementa
      aux+=a;
      if(a==0){
//! manda la suma total a padre y se cierra cuando recibe 0
        write(h1p[1],&aux,sizeof(int));
        exit(0);
      }
    }
  }else{
//! se crea el hijo 2
    if(fork()==0){
//! se cierran los canales
      close(h2p[0]);
      close(ph2[1]);
//! escucha indefinidamente
      while(1){
        read(ph2[0],&a,sizeof(int));
        printf("Soy hijo 2, recib  de padre: %d\n",a);
//! incrementa
```

```
51          aux+=a;
52          if(a==0){
53 //! manda a padre la suma total y se cierra cuando llega 0
54            write(h2p[1],&aux,sizeof(int));
55            exit(0);
56          }
57        }
58 //! continua el proceso original (padre)
59      }else{
60 //! cierra los canales que no se usan con su hijo, lectura y
       escritura
61        close(h1p[1]);
62        close(ph1[0]);
63        close(h2p[1]);
64        close(ph2[0]);
65        srand(time(NULL));
66        for(i=0;i<n;i++){
67 //! se hacen los randoms
68          aux=rand()%8+1;
69          if(aux%2==0){
70            printf("envi  a hijo 1: %d\n",aux);
71 //! si es par lo manda a hijo 1
72            write(ph1[1],&aux,sizeof(int));
73          }else{
74            printf("envi  a hijo 2: %d\n",aux);
75 //! si es impar lo manda a hijo 2
76            write(ph2[1],&aux,sizeof(int));
77          }
78        }
79 //! manda 0's para decirles que ya eso era todo
80        write(ph1[1],&leidos,sizeof(int));
81        write(ph2[1],&leidos,sizeof(int));
82        read(h1p[0],&a,sizeof(int));
83 //! imprime la suma total de hijo 1
84        printf("recib  la suma total de mijo 1: %d\n",a);
85        read(h2p[0],&b,sizeof(int));
86 //! imprime la suma total de hijo 2
87        printf("recib  la suma total de mijo 2: %d\n",b);
88      }
89    }
90    return 0;
91 }
```

Listing 1: Programa id de procesos y deteccion de señales del SO

Impresión de pantalla donde se puede apreciar el procedimiento gracias a los printf que facilitan la guia del proceso y aparte al final la suma total recibida por parte de los 2 hijos al padre.



Listings

5 Errores y problemas encontrados y como fueron resueltos

No se encontraron errores, y como tal no fue muy difícil la práctica lo que si hubo es mucha procrastinación por mi parte por ello se me hizo tardado.

6 Bibliografía

Apuntes en clase
https://ldc.usb.ve/ adiserio/ci3825/CLASEPS21.html
https://www.geeksforgeeks.org/pipe-system-call/
https://www.geeksforgeeks.org/c-program-demonstrate-fork-and-pipe/