Universidad Nacional Experimental Del Táchira
Departamento De Ingeniería En Informática
Sistemas Operativos Cod. 0425603t
Laboratorio 4 – Comunicación Interprocesos

Comunicación Inter-procesos

Tuberías En Linux

El principal objetivo de este laboratorio es comprender el funcionamiento de las tuberías, como usarlas para comunicar procesos entre sí y utilizar funciones para manejo de tuberías.

PARTE I. Uso de pipes desde la línea de consola

Ejecute los siguientes comandos:

```
$ ls > archivo
$ sort -r archivo
```

Aquí, salvamos la salida de **ls** en un archivo, y entonces ejecutamos **sort –r** sobre ese archivo para ordenarlo de forma invertida. Pero esta forma crea un archivo temporal para salvar los datos generados por **ls**. Para ahorrarse la necesidad de crear un archivo temporal se pueden los ***"pipes"*** o ***tuberías***. El uso de pipes es otra característica del intérprete de comandos, utilzada para conectar una cadena de comandos en un "pipe", donde la salida (stdout) del
primero es enviada directamente a la entrada (stdin) del segundo y así sucesivamente. En este caso, queremos conectar la salida de **ls** con la
entrada de **sort**. Para crear un pipe se usa el símbolo **"|"**:

```
$ ls | sort -r
$ ls /usr/bin | more
$ ps -ef | grep bash
```

PARTE II. Tuberías con y sin nombre

Las tuberías mencionadas, son unidireccionales. Solo pueden comunicarse en un solo sentido, sin embargo, podemos programar nuestras aplicaciones para crear tuberías multidireccionales. En lenguaje C utilizamos la función **pipe()** para crear nuestra tubería y enviar o recibir información a través de ella.

Información de pipe()

- **Descripción:** Crea un canal de comunicación entre procesos emparentados.
- **Parámetros:** Tabla para guardar los descriptores de entrada y de salida de la tubería.
- **Devuelve**: 0, si se ha completado correctamente; -1, en caso de error.

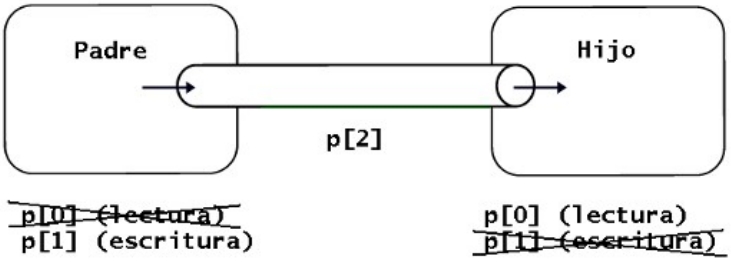
```
#include <unistd.h>
int descriptores[2];
int pipe (descriptores)
```

Comentarios:

- descriptores[0] se abre para lectura y descriptores[1], para escritura.
- La operación de lectura en descriptores[0] accede a los datos escritos en descriptores[1] como en una cola
- FIFO (primero en llegar, primero en servirse).

Ejemplo en Lenguaje C

En el siguiente diagrama, observamos una tubería conectada entre dos procesos, consiste en un arreglo de 2 posiciones donde en la primera se utiliza para leer y la otra para escribir información en ella.




```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#define LEER 0
#define ESCRIBIR 1

int main () {
   int descr[2]; /* Descriptores de E y S de la tubería */
   int bytesleidos;
   char mensaje[100],
   *frase="EL padre prueba al hijo si la transferencia es buena.";

   printf ("Ejemplo de tubería entre padre e hijo.\n");
   pipe (descr);

   if (fork () == 0) {
      close (descr[LEER]);
      write (descr[ESCRIBIR], frase, strlen(frase));
      close (descr[ESCRIBIR]);
   } else {
      close (descr[ESCRIBIR]);
      bytesleidos = read (descr[LEER], mensaje, 100);
      mensaje[bytesleidos]='\0';
      printf ("Bytes leidos: %d\n", bytesleidos);
      printf ("Mensaje del Padre: %s\n", mensaje);
      close (descr[LEER]);
   }
   return 0;
}
```

PARTE III. Asignación

1. Modificar el ejemplo anterior para realizar comunicación en ambos sentidos utilizando tuberías, debe especificar en los mensajes el sentido para facilitar al usuario quien escribió el mensaje y quien lo recibió.
2. Investigar sobre el uso del comando **mkfifo**, realice un *programa o secuencia de comandos* para listar los archivos del directorio actual en forma descendente, y permita a través de un parámetro escribir una cadena de texto para filtrar los resultados. El *programa o secuencia de comandos* debe:
    - Utilizar tuberías con nombre usando mkfifo.
    - Si lo hace mediante un script utilice los comandos **ls, sort y grep** para facilitar el proceso.
    - Valide el número de parámetros recibidos.

Valor: 5 puntos cada una de las 2 asignaciones.