Licenciatura en Computación
PROGRAMACIÓN CONCURRENTE
Práctica 6: Tuberías o Pipes

Trimestre 23i

1. Objetivos

a. Comprender el funcionamiento de los pipes.
b. Identificar las situaciones en las que son útiles.
c. Realizar ejercicios utilizando intercambio de mensajes entre procesos usando pipes nombrados y sin nombrar.

2. Introducción:

Las tuberías son una de las primeras formas de comunicación implantadas en UNIX y en muchos de los sistemas actuales se cuenta con está utilería para comunicar procesos.

Una tubería se puede considerar como un canal de comunicación entre dos procesos y las hay de dos tipos: tuberías con nombre (fifos) y tuberías sin nombre.

Tuberías sin nombre: las tuberías sin nombre se crean con la llamada `pipe` y sólo el proceso que hace la llamada y sus descendientes pueden utilizarla. La llamada tiene la siguiente signatura:

```
#include <unistd.h>

int pipe(int fildes[2]);
```

Si la llamada funciona correctamente devolverá el valor de 0 y creará una tubería sin nombre; en caso contrario, devolverá -1 y dejará en errno el código de error producido. La tubería creada se puede utilizar a través del contenido en `fildes[2]`. Los dos elementos se comportan como dos descriptores de archivo y se pueden usar para escribir en la tubería y leer de ella. Al escribir en `fildes[1]` se introducen datos en la tubería y al leer de `fildes[0]` se extraen datos de ella, por lo tanto `fildes[1]` se comporta como un archivo de sólo escritura y `fildes[0]` como archivo de sólo lectura. El kernel asigna un inodo a la tubería y lo trata como un archivo manipulable a través del par de descriptores de archivo (`fildes[0]` y `fildes[1]`).

Para poder leer y escribir en una tubería se usan las mismas llamadas a `write` y `read` usadas para archivos y directorios.

```
#include <unistd.h>
ssize_t read(int  fdes, void *buf, size_t  nbyte);
ssize_t write(int  fdes, const void *buf, size_t  nbyte);
```

Donde la función `write` intenta escribir *nbyte* bytes en el buffer con dirección *buf* al archivo asociado al descriptor de archivo abierto *fdes*. En el caso del pipe este parámetro debe ser el extremo 1 del arreglo `fildes`.

Por otra parte la función `read` intenta leer *nbyte* bytes del archivo abierto con descriptor *fdes* y los almacena en el buffer cuya dirección es *buf*. En el caso del pipe este parámetro debe ser el extremo 0 del arreglo `fildes`.

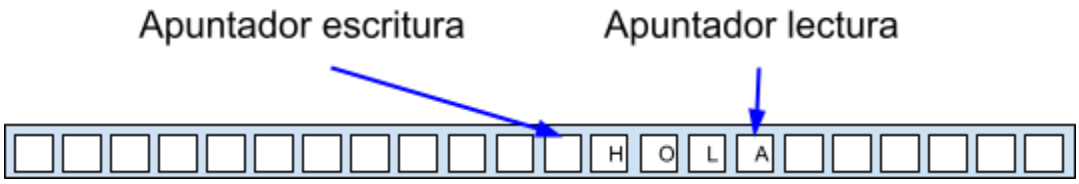
Para mayor información se recomienda consultar el manual correspondiente.

Tuberías con nombre (FIFO)

Para los procesos que no guardan ninguna relación entre ellos es necesario recurrir a las tuberías con nombre. Un fifo es un archivo con una semántica idéntica a la de una tubería sin nombre, pero ocupa una entrada en un directorio y se accede a él a través de un nombre.

Un proceso puede abrir una tubería con un nombre mediante una llamada a open de la misma manera como lo hace con un archivo, por lo cual para comunicar dos procesos mediante un fifo, uno de ellos debe de abrir la tubería para escribir y el otro para leer. Cuando un proceso abre un fifo para escribir en él, se pone a dormir hasta que no haya otro proceso que lo abra para leer de él y de forma inversa cuando algún proceso lo abre para leer se pone a dormir hasta que algún proceso lo abre para escribir.

Para controlar los accesos de lectura y escritura del fifo, el kernel emplea dos apuntadores. Los bloques directos de direcciones del `inode` son manejados como si fuesen nodos de una cola circular, de tal forma que cuando el puntero de escritura llega al último de los bloques, empieza por el primero, y lo mismo ocurre para el puntero de lectura. Ambos punteros son gestionados del tal forma que el acceso al fifo es de tipo *first-in-first-out*.



Para crear una tubería nombrada portable se usa la llamada a mknod:

#include <sys/stat.h>

int mknod(const char *`path`, I_FIFO|Bandera, 0);

Que tiene el efecto de crear un archivo con el camino/nombre indicado en el primer parámetro. El segundo parámetro indica que se usará como FIFO con los permisos indicados en `bandera` (usualmente se utiliza `0666` indica que el derecho de acceso en lectura y escritura para todos).

3. Actividades

A. Compilar, ejecutar y comprender los programas ejemplo que se encuentran en el aula virtual.
B. Deduzca la capacidad de un pipe.
C. Modifique el programa ejemplo para que el proceso emisor le envíe varios mensajes al proceso receptor.
D. Agregue lo que sea necesario para que la comunicación pueda ser en los dos sentidos.
E. Elabore una nueva versión del programa para que dada una secuencia de números, encuentre aquel número cuya secuencia de Collatz sea la más larga. En esta nueva versión utilice procesos y pipes en lugar de hilos y memoria compartida.

4. Entregables: Elaborar un reporte en pdf en donde para los incisos B,C,D y E se explique la solución, se incluya el código y el enlace a gdb.

5. Fecha de entrega: La indicada en el aula virtual.

Elaboró: Elizabeth Pérez Cortés