FIFOS

FIFOS – tuberías con nombre

- Un FIFO es similar a un pipe, con la diferencia es que un FIFO tiene un nombre en el sistema de archivo.
- Se abre de la misma manera que un archivo normal.
- Se pueden abrir configurados como bloqueantes o no bloqueantes.
- Puede ser utilizado para la comunicación entre procesos no relacionados.
- Una vez que el FIFO se ha creado, cualquier proceso puede abrirlo.
- Utiliza las mismas funciones que utilizamos con pipe y otros archivos: read(), write() y close().
- Tiene un extremo de escritura y uno de lectura.
- Cuando **todos los descriptores de un FIFO se cierran**, l**os datos pendientes se descartan**. El FIFO permanece.
- Los **datos** tienen **persistencia de proceso**.

Creación de un FIFO

La función para crear un FIFO es mkfifo:

```
#include <sys/stat.h>
int mkfifo(nombre, mode_t mode);
```

Devuelve 0 si tuvo éxito o -1 en caso de error.

En el campo mode se colocan los permisos del FIFO (lectura, escritura, ejecución) en octal.

Creación de un FIFO

Podemos crear un FIFO desde el shell con el comando mkfifo:

```
$ mkfifo [ -m mode ] pathname
```

El pathname es el nombre de la FIFO.
La opción -m se utiliza para especificar los permisos.

Cuando listamos con el comando ls -l un FIFO se muestra con un tipo p en la primer columna.

Para borrar el FIFO desde un shell usamos el comando rm.

Apertura de un FIFO open()

La llamada al sistema open() abre un archivo existente o crea y abre un nuevo archivo.

```
#include <sys/stat.h>
#include <fcntl.h>
int open(nombre, int flags, ... /* mode_t mode */);
```

Devuelve el descriptor del archivo si tuvo éxito o -1 en caso de error.

El campo mode indica los permisos al crear un archivo, en este caso este campo ser coloca en cero. El campo flag indica el modo de apertura:

O_RDONLY: Abre el archivo sólo para lectura

O_WRONLY: Abre el archivo para sólo escritura

O_RDWR: Abre el archivo para lectura y escritura

O_NONBLOCK Abre en modo sin bloqueo

Podemos abrir el FIFO de modo bloqueante o No bloqueante

- Un FIFO creado puede ser abierto por cualquier proceso (si tiene los permisos).
- La apertura de una FIFO para lectura (flag O_RDONLY en open()) bloquea al procesos hasta que otro proceso abra el FIFO para escritura (flag O_WRONLY en open()).
- De la misma forma, la apertura de la FIFO para escribir bloquea al proceso hasta que otro proceso abre el FIFO para lectura.
- Por lo dicho en el parrafo anterior la apertura de una FIFO sincroniza los procesos de lectura y escritura.
- Si queremos que el proceso no se bloquee debemos colocar el flag: O_NONBLOCK en la llamada open(), o utilizar en la función open() el flag O_RDWR (esta opción no es la mas recomendada).

Borrar un FIFO

```
#include <unistd.h>
int unlink( const char *pathname);
```

Devuelve 0 si tuvo éxito, o –1 en caso de error.

Bibliografía

Kerrisk, Michael. *The linux programming Interface.* 2011. **Capítulo 44.**