Info_sw
www.wuolah.com/student/Info_sw

APSO_apuntes_practica5.pdf

APSO. Parte 2: programación

2° Administración y Programación de Sistemas Operativos

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería
UHU - Universidad de Huelva

APSO. PROGRAMACIÓN DEL SISTEMA.

APUNTES PRÁCTICA 5:

COMUNICACIÓN ENTRE PROCESOS

********** FIFOS **********

Las FIFOS son ficheros especiales (un proceso escribe lo que sea y si otro proceso lo lee, se extrae

del fichero y no permanece en él como en un fichero normal).

ESTO LO HARÁ EL PROCESO QUE SE ENCARGUE DE CREAR LA FIFO, QUE PUEDE SER ALGUNO DE LOS PROCESOS QUE

LEAN O ESCRIBAN EN ELLA U OTRO TERCER PROCESO QUE NO LEA NI ESCRIBA (LO ESPECIFICARÁ EL ENUNCIADO):

mkfifo() -> sirve para crear un fichero de tipo FIFO. Solo ejecutará este comando uno de los procesos.

SINTAXIS: mkfifo("nombre_fifo",permisos_del_fichero);

EJEMPLO: mkfifo("fifo1",0777); //permisos en binario (con un '0' delante -¡¡no olvidar!!-).

unlink() -> sirve para borrar la FIFO (o cualquier fichero)

EJEMPLO: unlink("fifo1");

//MUY IMPORTANTE: TRAS LA EJECUCIÓN DEL PROGRAMA DEBEN

//ELIMINARSE LAS FIFOS!! -> En el examen resta bastante si no se eliminan!!

ESTO LO HARÁ TANTO EL PROCESO QUE LEE COMO EL QUE ESCRIBE EN LA FIFO:

Una vez creada, para trabajar con ella habrá que abrirla (OJO: abrir SIEMPRE las FIFOS en modo O_RDWR

para evitar problemas como el que detallamos más adelante).

```
int fd;
fd=open("fifo1",O_RDONLY); //ABRIMOS LA FIFO PARA LECTURA
fd=open("fifo1",O_WRONLY); //ABRIMOS LA FIFO PARA ESCRITURA
fd=open("fifo1",O_RDWR); //ABRIMOS LA FIFO PARA LECTURA Y ESCRITURA
```

Para leer/escribir se utiliza el read y write:

```
read(fd,&variable,tamaño_en_bytes); //para el tamaño usar 'sizeof(variable)'
write(fd,&variable,tamaño_en_bytes); //para el tamaño usar 'sizeof(variable)'
```

IMPORTANTE: Si un proceso hace read de una FIFO que no tiene nada, el proceso se queda parado

esperando hasta que en la FIFO haya algo. Si no hay ningún otro proceso que tenga la FIFO abierta

para escritura, entonces el read se lo salta, y entonces no lee (cuidado con hacer un read antes de

un open en modo escritura. En el caso contrario, pasa algo parecido o incluso peor (ver apuntes).

-> Asegurarnos antes de hacer un read o un write que la FIFO esté abierta en ambos modos!!

-> Para evitar estos errores la solución es abrir SIEMPRE la FIFO en modo lectura y escritura (O_RDWR)

Tras leer/escribir, habrá que cerrar la FIFO (esto no lo explicó en clase, igual es porque no es necesario):

close(fd); //CREO QUE ESTO NO ES NECESARIO CUANDO ABRIMOS LA FIFO EN MODO "O_RDWR". AL MENOS

//EN LA PRÁCTICA 5 NO FUNCIONABA PONIENDO EL CLOSE() Y AL QUITARLO YA FUNCIONABA.

********** TUBERÍAS **********

Las tuberías son unos ficheros especiales.

Para poder crear una tubería, previamente debemos crear un array de 2 posiciones:

int tubo[2];

Ahora ya podremos crear una tubería, pasándole el array creado como parámetro:

pipe(tubo); //coge las dos primeras posiciones libres de la tabla de canales. Normalmente las

//posiciones 0, 1 y 2 ya están ocupadas y cogerá normalmente la 3 y 4.

//Las posiciones escogidas se guardan en el array de dos posiciones.

//tubo[0] guarda la posición del puntero de lectura en la tabla de canales

//tubo[1] guarda la posición del puntero de escritura en la tabla de canales

TABLA DE CANALES

+----------+

0 | TECLADO (entrada estandard) |

+----------+

1 | PANTALLA (salida estandard) |

+----------------------------+

2 | PANTALLA (salida de errores)|

+----------------------------+

3 | Puntero lectura tubería | <- TUBO[0]=3

+----------------------------+

4 | Puntero escritura tubería | <- TUBO[1]=4

+----------------------------+

5 | ... |

+----------------------------+

6 | |

Tras hacer esto, la tubería ya está abierta (no hay que hacer 'OPEN' como en las FIFOS).

Para leer/escribir haremos:

```
read(tubo[0],&variable,tamaño_en_bytes); //para el tamaño usar 'sizeof(variable)'
write(tubo[1],&variable,tamaño_en_bytes); //para el tamaño usar 'sizeof(variable)'
```

Ojo que el execl() solo hereda las 3 primeras posiciones de la tabla de canales. Habrá que modificar

la tabla de canales (en concreto se recomienda modificar la posición 2 o salida de errores):

Antes de hacer el execl() en la parte de la copia realizada por el fork() haremos:

```
close(2);    //cerramos la entrada 2 de la tabla de canales y la dejamos libre.
```

dup(tubo[0]); //duplicamos el puntero de lectura (tubo[0]) o de escritura (tubo[1]) en la

//posición que hemos cerrado previamente. 'dup' siempre duplica en la primera entrada

//libre que encuentre en la tabla de canales (si cerramos la 2, esta será la elejida).

OJO CON ESTO: En el proceso hijo, al leer/escribir NO podremos hacerlo con tubo[0] o tubo[1], sino con

la posición que hemos cerrado de la tabla de canales (en nuestro caso la posición 2):

read(2,&variable,tamaño_en_bytes); //para el tamaño usar 'sizeof(variable)'

write(2,&variable,tamaño_en_bytes); //para el tamaño usar 'sizeof(variable)'
