Sistemas Operativos - Comunicación entre procesos (2)

-----------------------------------------------------

Comunicacion sencilla entre procesos por medio de tuberias

(pipes) con nombre y sin nombre.

Las tuberias (pipes) son 'conductos' unidireccionales por las cuales podemos hacer

circular informacion.

+------+

A --> | pipe | --> B

+------+

Estos 'conductos' existen en el espacio de memoria del SO y son provistos

por el.

Desde el punto de vista de los procesos no se pueden diferenciar de

leer o escribir en un archivo.

Existen de dos tipos:

\* con nombre (existe 'visible' en el sistema de archivos)

\* sin nombre (existe solo en el espacio de memoria de los procesos)

Las tuberías 'con nombre' pueden comunicar procesos NO relacionados

(padre, hijo, hermano)

Las tuberías 'sin nombre' solo pueden ser utilizadas por procesos

relacionados (padre, hijo, hermano) ya que solo ellos pueden 'saber'

como acceder a ellos

Las tuberías 'con nombre' se crean por medio del comando mkfifo,

ej:

$ mkfifo test

$ ls -l test

prw-rw-r-- 1 bgkriegel bgkriegel 0 abr 17 06:42 test

Note la primer letra, que indica tipo, su valor es 'p' de 'pipe'

Las tuberías 'sin nombre' se crean por medio de la llamada al sistema

pipe(), que retorna un arreglo de dos descriptores (como los de archivos),

con la particularidad que cada uno apunta a uno de los extremos de la tubería,

pipe[0] es la punta de lectura

pipe[1] la punta de escritura

+------+

pipe[1] (write) --> | pipe | --> pipe[0] (read)

+------+

en código:

int pd[2]; /\* declaro \*/

pipe(pd); /\* creo \*/

read( pd[0], buff, max ); /\* leo \*/

write( pd[1], buff, max ); /\* escribo \*/

NO se puede leer y escribir en el mismo proceso!!

Ejemplos:

1) Con nombre

\* Crear una tubería como se explico, tub1

\* Se usan 3 consolas, A y B

\* en la consola A ejecute: $ "ls -l / > tub1" (observe como se bloquea)

\* en la consola B ejecute: $ "cat < tub1" (verá la salida)

2) Sin nombre

\* ejecute: "ls -l / | cat"

2 hace lo mismo que 1, pero en 2 la tubería es creado por ls shell, hace 2 hijos

para ejecutar los comandos y cada hijo le 'pasa' la salida al otro utilizando una

tubería sin nombre

Ejercicios:

1) Cliente-Servidor con archivos. Exploraremos la posibilidad de "comunicar"

dos procesos mediante archivos externos. Escriba un programa que "cree" dos

archivos, a-inout y b-inout. Luego deberá crear dos hijos, cada hijo deberá

abrir los archivos PERO de modo que el primer hijo abra "a" para lectura y

"b" para escritura, en cambio el segundo abrira "a" para escritura y "b"

para lectura. Una vez realizado esto el primer hijo debera obtener la hora

del sistema (usando ctime()), escribira esta informacion en "b", cuando esto

ocurra, el segundo proceso hijo obtendra a su vez la hora y la leerra de

"b", y la mostrara por pantalla. Al finalizar ambos procesos hijos

el padre deberá reportar el estado de terminacion.

3) Cliente-Servidor con pipes sin nombre. Escribiremos un programa que

creara dos hijos los cuales se comunicaran utilizando pipes sin nombre.

Uno de los hijos obtiene la hora del sistema como antes, y envía ese dato

utilizando pipes. El segundo recibe los datos y los muestra por pantalla.

3) Idem anterior (en cuanto a funcionalidad) pero usando dos programas

diferentes en consolas diferentes y con un pipe con nombre.