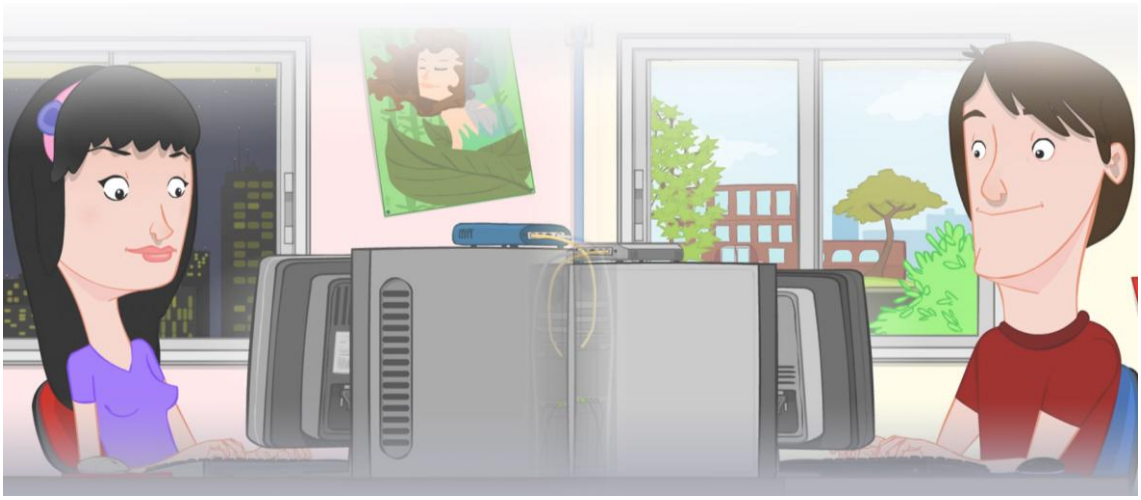
Práctica 2

Comunicación entre procesos de la misma máquina



Curso 2013-2014

Grado en Ingeniería Técnica de Telecomunicación

(3º curso)

Área de Ingeniería Telemática

Práctica 2: Comunicaciones entre procesos de la misma máquina

1. Motivación:

A la hora de intercambiar datos entre procesos que residen en la misma máquina, existen en UNIX una serie de recursos de comunicaciones clásicos que conviene que todo alumno conozca para poder utilizarlos en el diseño de aplicaciones más complejas. Entre estas soluciones de comunicaciones entre procesos (IPCs) se encuentran los pipes que es el sistema original de UNIX que permite comunicar dos procesos que son de la misma jerarquía, es decir que nacen del desdoble con fork() del mismo programa en ejecución. Muchas veces, necesitamos intercambiar datos entre procesos que no son de la misma jerarquía (es decir que ejecutan programas diferentes). Para ello es necesario disponer de un punto de encuentro común que nos permita identificar un recurso que puedan compartir ambos procesos para comunicarse. Es el caso de los named-pipes, que utilizan un fichero intermedio (en memoria) de intercambio de datos localizado en el sistema de ficheros, o el de las colas de mensajes del sistema, que propone un mecanismo de almacenamiento de mensajes por el sistema que puede ser utilizado eficazmente por los procesos para intercambiar información. Esta serie de recursos son el objeto de trabajo de esta práctica. Hay en internet infinidad de ejemplos para su uso. Conociendo estos recursos se pueden resolver problemas prácticos de comunicaciones entre procesos dentro de una misma máquina. En la segunda parte de la práctica se propone al alumno que diseñe una aplicación de comunicaciones de mensajes un poco más completa.

2. Enunciado de la práctica

Esta práctica tiene un tiempo de desarrollo de tres semanas y está organizada en dos pruebas.

En la primera, los alumnos deberán diseñar y escribir una pequeña aplicación en la que se utilizarán los recursos de comunicación entre procesos de la misma máquina presentados en las clases teóricas. Basándose en los requerimientos de la aplicación el alumno deberá determinar las posibles opciones de diseño e implementación, pero junto con el enunciado se aportan sugerencias para adquirir los contenidos que se deberían de cubrir con esta práctica.

En el segundo apartado se proponen algunos ejercicios guiados en los que el alumno realizará unos programas que se comunicarán con una aplicación de evaluación (el *monitor*). Este apartado permitirá al alumno comprobar su nivel de dominio de los conceptos y habilidades que se trabajan en esta práctica.

2.1 Descripción de la primera prueba práctica

2.1a.-Comunicaciones entre procesos

Caso de comunicación entre procesos de la misma jerarquía

Crea un programa cuyo proceso padre se desdoble en dos procesos hijos que utilicen un pipe para comunicarse. El pipe servirá para mandar mensajes del proceso hijo 1 al hijo 2. El proceso hijo 1 recogerá mensajes del teclado y utilizando el pipe se los mandará al otro proceso hijo 2 que traducirá el mensaje recibido a mayúsculas y lo presentará en pantalla.

Caso de comunicación entre procesos independientes (Cliente/Servidor)

Ahora se trata de comunicar dos procesos que ejecutan dos programas diferentes. Uno como cliente y otro como servidor. EL cliente debe recoger un mensaje del teclado y enviarlo al proceso servidor a través de uno de los recursos de comunicaciones propuestos. El servidor recogerá el mensaje a través de ese recurso y después de procesarlo (p.e. traducirlo a mayúsculas) devolvérselo al proceso cliente por el mismo u otro recurso de comunicaciones entre procesos.

En un caso se propone utilizar named-pipes como recurso de comunicaciones.

En otro caso se propone utilizar las colas de mensajes del sistema como recurso de comunicaciones.

Una vez implementados los diferentes mecanismos conviene observar las siguientes cuestiones:

- ¿Qué ventajas tiene un mecanismo sobre otro (pipes vs named-pipes vs mensajes)?
- ¿Cómo se han creado los named-pipes?
- ¿Qué cosas hay que tener en cuenta al utilizar los named-pipes en cuanto a la forma de abrirlos?
- ¿Qué límites imponen la comunicación entre procesos a través de pipes, named-pipes o colas de mensajes?
- Busca con utilidades del sistema los recursos (named-pipes, colas de mensajes) que crees que está manejando el S.O. y comprueba su estado.

2.1b.- Diseño de una aplicación (Cliente/Servidor) con múltiples clientes

Generalizar el ejercicio anterior utilizando colas de mensajes para la comunicación entre diferentes procesos independientes, construyendo un proceso servidor que atienda mensajes de varios procesos cliente independientes arrancados por diferentes usuarios y proporcione un servicio de traducción de mensajes de minúsculas a mayúsculas.

El proceso servidor recibirá los mensajes de los diferentes procesos cliente a través de una cola de mensajes determinada. Procesará cada mensaje y contestará a cada proceso cliente a

través de la cola de mensajes. Será el proceso cliente quién mostrará en pantalla el mensaje traducido.

Para poder simular el trabajo de varios procesos clientes de forma simultánea, conviene modificar el programa cliente para permitir el envío de mensajes de forma desatendida. Es decir en vez de leer los mensajes desde el teclado, permitir que algunos procesos lancen un número determinado de mensajes de una lista proporcionada, guardando una temporización definida a gusto del usuario entre mensajes. Se debe de conservar también el procedimiento original por si se quisiera realizar pruebas manuales.

A la hora de pensar en el diseño de la aplicación conviene analizar algunos aspectos como:

- ¿Cómo se distinguen los mensajes de cada proceso cliente en la cola?
- ¿Cómo conoce el proceso servidor donde mandar los mensajes a los procesos cliente?
- ¿Cómo se gestiona la entrega de mensajes de respuesta hacia los procesos cliente?
- ¿En qué estado quedan las colas de mensajes?

Competencias a desarrollar por el alumno

1. Diseño de un sistema de comunicaciones entre procesos. Gestión de recursos.
2. Manejo de los recursos clásicos de comunicaciones entre procesos: pipes, named-pipes, colas de mensajes.
3. Organización de una aplicación: Introducción de las directivas de pre-procesamiento.
4. Fases en la compilación: Precompilar-Compilar-enlazar (linkar).
5. Llamadas al sistema bloqueantes: msgrcv(), read(), ….
6. Comandos básicos de gestión de recursos de comunicaciones IPCs: mkfifo(), mknod(), ipcs,ipcrm, …

Guía de desarrollo de la práctica

El desarrollo de la práctica en todo su enunciado dependerá del interés del alumno y del trabajo adicional que pueda realizar fuera de las horas regladas de laboratorio. Esta guía está más enfocada a proponer una metodología sencilla de trabajo en laboratorio que pueda servir para formar suficientemente al alumno para conseguir sus objetivos. La secuencia de pasos a realizar en el tiempo de laboratorio es la siguiente:

1. Pensar bien en la reutilización del código que se está desarrollando pues si el código que se realiza es suficientemente flexible, los únicos cambios que hay de un programa a otro afectan sólo al tipo de recurso de comunicaciones que se usa. La lógica de cada programa es similar. Es una buena ocasión para aprender a usar las instrucciones de preprocesamiento, saber generalizar funciones, crear librerías y utilizar makefiles.
2. El primer caso de pipes es un poco especial y se puede hacer aprovechando la creación de jerarquía de procesos de la práctica anterior. Lo más delicado es saber crear y usar de forma adecuada el recurso de pipe generado con la primitiva pipe(). Es un recurso con dos

descriptores de dispositivo que tiene un funcionamiento peculiar. Hay infinidad de ejemplos en internet sobre su uso. Pensar en dónde debe crearse el recurso pipe dentro de la jerarquía para que los procesos que se quieran comunicar sepan acceder correctamente al recurso.

3. Los demás ejercicios se pueden hacer con una estructura de programa similar tanto para el cliente como para el servidor. Son dos programas diferentes pero ambos deben de identificar el recurso común de comunicaciones. En UNIX es común acceder a los recursos externos como si fuesen dispositivos. Eso es aplicable también a los ficheros. Eso permite que las aplicaciones de UNIX puedan fácilmente comunicarse. Un consejo es que creéis en vuestros programas funciones suficientemente genéricas tanto para recibir datos como para enviarlos que puedan personalizarse para cada caso según usemos un recurso u otro.
4. En el caso de named-pipes, el recurso a usar es un recurso externo que debe existir en el sistema de ficheros. Hay dos opciones: (a) crearlo desde programa (mkfifo()), o (b) crearlo desde el sistema (comando mknod). Aparece como si fuese un fichero pero realmente es un dispositivo especial. Con ls –l podéis verlo y el sistema os dará información de que tipo de dispositivo es. Cuidado con los permisos.
5. La forma de trabajo con un dispositivo named-pipe es similar a como se hace con dispositivos o ficheros; usando las primitivas open(), write(), read(), close(). Si revisáis el manual de open() (man 2 open) veréis que opera de una forma especial con dispositivos de este tipo para sincronizar los procesos. Por ejemplo si un proceso intenta abrir el recurso para leer, el sistema lo retiene hasta que no haya otro proceso que abra ese recurso para escribir. Como esto muchas cosas más que podéis aprender leyéndoos las páginas de manual de las funciones que usáis.
6. En el caso de las colas de mensajes, la creación del recurso se debe hacer desde la aplicación a través de las primitivas msgget() con las claves y permisos adecuados. El envío de mensajes y recepción se hace con primitivas específicas como msgsnd() y msgrcv(). SI creáis bien la cola de mensajes la podéis ver a través del comando ipcs.
7. Un primer problema lo tenéis a la hora de identificar el recurso que queréis usar. Se utiliza una clave que se supone que es única y que nadie que no sea vuestra aplicación va a utilizar la misma clave. Para ello conviene que elijáis una clave única. Estas claves están en formato long y se suelen representar en Hexadecimal. Si usáis ipcs para ver lo que habéis creado conviene que uséis ese formato para generar las claves. Una idea es hacer un #define MI_CLAVE 0x1234AFL. La L del final es importante para que el compilador sepa que es un long. En UNIX hay un procedimiento pseudo-estándar de generación de claves supuestamente únicas a través de ftok() y usando como identificador un fichero de nuestra aplicación. Su uso no es obligatorio pero si interesante saber en qué consiste.
8. Otro problema a solventar con las colas de mensajes es entender como estructura los mensajes el sistema. Si creamos una cola de mensajes para nuestra aplicación, a través de esa cola podemos dejar mensajes de diferente tipo y los procesos que quieren leer mensajes pueden “sintonizarse” (es decir leer) mensajes de un tipo determinado. De esta forma podemos mantener varias comunicaciones simultáneas a través de la misma cola de mensajes.
9. Habrá que decidir qué tipo de mensajes usar para cada comunicación y quienes intervienen en ello. Los tipos de mensaje es algo que forma parte del propio mensaje y que

hay que saber utilizar adecuadamente. Un proceso receptor de mensajes puede "sintonizarse" en un tipo determinado vía msgrcv().

10. Se aconseja usar etiquetas (flags) de compilación para decidir sobre el código de cada programa que versión compilar; si named-pipes o mensajes. De esta forma cualquier modificación en la lógica general se mantendrá actualizada para ambos casos y nos podemos centrar en cada caso en la forma de generalizar el uso de cada recurso. Se pueden activar estos flags de compilación desde el make.
11. El diseño de la aplicación final que se propone de traducción de mensajes exige una reflexión previa de cómo utilizar la cola de mensajes. Un problema importante a resolver es cómo puede saber el servidor por dónde dar la respuesta al cliente. Aquí el alumno debe plantear alguna solución que seguramente le hará reflexionar sobre el concepto de "protocolo".

2.2 Descripción de la segunda prueba práctica

El alumno debe realizar un programa denominado <client.c> que se ejecutará como <./client> en el directorio de trabajo donde reside el programa <./monitor2>. El programa monitor propone la realización de 7 ejercicios que pueden ejecutarse de forma independiente pero que están enunciados pensando en que se hagan de forma progresiva.

Ejercicio 1: Leer mensajes de un named-pipe

El programa monitor creará, en el sistema de ficheros, un dispositivo tipo fifo (fifo1) denominado "/tmp/fifo_monitor_1" y esperará durante un tiempo a que el proceso cliente abra el dispositivo para realizar una lectura de un mensaje. Cuando el monitor detecte que el cliente ha abierto el dispositivo procederá a escribir un mensaje de texto con formato "<999>" donde indicará el secreto <1> que el alumno debe desvelar. El proceso monitor no conocerá de momento si el proceso cliente ha identificado adecuadamente el secreto pero en el ejercicio 2 lo podrá descubrir. Para ello el proceso cliente debe guardar el secreto obtenido y mantener el fifo1 abierto disponible para su uso.

Ejercicio 2: Escribir mensajes en un named-pipe

El proceso monitor esperará durante un tiempo a que el programa cliente cree un dispositivo fifo (fifo2) denominado "/tmp/fifo_monitor_2" que será utilizado para mandar mensajes del cliente al monitor. Cuando aparezca ese recurso el proceso monitor lo abrirá para leer mensajes de él. El primer mensaje que se espera recibir es un texto con formato "<999>" donde se indique el secreto <1> obtenido en el primer ejercicio. Al recibir este mensaje el monitor resolverá si es correcto o no el secreto <1> y enviará por el fifo1 ("/tmp/fifo_monitor_1") un mensaje con formato "<999>" indicando el secreto <2>. El programa cliente tendrá que leer ese mensaje y responder en eco por el fifo2 ("/tmp/fifo_monitor_2") de forma inmediata con el mismo formato "<999>" el secreto <2>.

Ejercicio 3: Leer mensajes de una cola de mensajes según SYS V

El proceso monitor creará una cola de mensajes según el estándar de SYS V para intercambiar mensajes con el proceso cliente. Esta cola de mensajes tendrá por clave de acceso un número long que coincidirá con la representación en hexadecimal de la parte numérica del DNI del alumno. Es decir que si el DNI es "12345678G", el hexadecimal que representa su clave es 0x12345678L (la L final informa al compilador que se debe representar este número como un long en memoria). El monitor enviará un mensaje a esa cola cuyo tipo coincidirá con la clave secreta <3> y cuyo contenido indicará la clave secreta <4> según el formato "<999>". El proceso cliente debe guardar estos dos secretos para comunicarlos posteriormente.

Cada vez que el alumno seleccione la opción 3, el monitor eliminará de la cola de mensajes todos los mensajes y procederá a escribir de nuevo en la cola un sólo mensaje con los secretos <3> (en el tipo de mensaje) y <4> (en el contenido con formato "<999>").

Ejercicio 4: Escribir mensajes en una cola de mensajes (SYS V)

El proceso monitor destruirá (si existe) la cola de mensajes que tiene como clave el DNI del alumno en hexadecimal (Ej: 0x12345678L). Debe ser el cliente quién construya el recurso de nuevo con esa clave para poder comunicarse con el monitor. El proceso monitor testeará durante un tiempo determinado la existencia de la cola de mensajes con la clave del alumno y cuando detecte su presencia procederá a leer un primer mensaje que el proceso cliente debe introducir en esa cola. No importa el tipo de mensaje que se utilice. El contenido tendrá el formato "<999><888>" donde 999 codificará el secreto <3> y 888 el secreto <4>. Al recibir esto, el monitor indicará si el descubrimiento de cada secreto es correcto o no. Si, al menos, el formato del mensaje es adecuado, independientemente que el secreto <3> y <4> estén bien identificados, el proceso monitor desvelará el secreto <5> en pantalla.

Ejercicio 5: Leer mensajes de determinado tipo de una cola de mensajes (SYS V)

Al seleccionar esta opción, el proceso monitor comprobará si existe la cola de mensajes con la clave del alumno. Si existe, eliminará todos los mensajes pendientes de la cola de mensajes. Si no existe creará la cola de mensajes de nuevo. Posteriormente, el proceso monitor introducirá una serie de mensajes de diferente tipo en la cola de mensajes. Sólo uno de esos mensajes tendrá como tipo un long que coincidirá con el pid del proceso cliente y llevará como contenido una cadena con formato "<999>" donde estará codificado el secreto <6>. Se espera que el cliente sea capaz de recoger ese mensaje únicamente de la cola y que guarde el secreto para comunicárselo al monitor en el ejercicio siguiente.

Ejercicio 6: Escribir mensajes de determinado tipo en una cola de mensajes (SYS V)

El proceso monitor comprobará si existe la cola de mensajes con la clave correspondiente al DNI del alumno. SI no existiese la creará. Posteriormente esperará a recibir un mensaje del tipo correspondiente al pid del proceso monitor con el formato "<999>" en su contenido. Si recibe este mensaje con el formato adecuado descubrirá el secreto <7> dando por entendido que el alumno sabe escribir mensajes en la cola de determinado tipo. Si además el valor numérico contenido en el mensaje (999) coincide con el secreto <6> dará por buena la realización del ejercicio 5. Es decir, el mensaje que debe enviar el proceso cliente debe ser del tipo correspondiente al pid del proceso monitor, y debe codificar el secreto <6> en el contenido del mensaje con formato "<999>".

Ejercicio 7: Eliminar los recursos de comunicaciones IPCs creados

Se espera que el proceso cliente, sea capaz de eliminar los recursos utilizados. Cuando se active la opción, el proceso monitor comprobará que los recursos están disponibles y esperará un mensaje cualquiera del cliente a través de la cola de mensajes con tipo el pid del monitor. Este mensaje servirá de inicio de la prueba. Unos segundos después, el monitor, comprobará que el proceso cliente ha eliminado del sistema los recursos. Si es así, descubrirá el secreto <8>.

PRUEBA LABORATORIO DE ARQUITECTURA DE SISTEMAS DE LA INFORMACIÓN

Nombre:…… Grupo: ……

DNI:…… Fecha:……

Ejercicio 1:Leer de fifo Secreto 1: Ejercicio 2: Escribir en fifo Secreto 2:

Ejercicio 3: Leer de cola de mensajes Secreto 3: Secreto 4:

Ejercicio 4: Escribir en cola de mensajes Secreto 5: Ejercicio 5: Leer un tipo de mensaje Secreto 6:

Ejercicio 6: Escribir un tipo de mensaje Secreto 7:

Ejercicio 7: Eliminar los recursos Secreto 8: Firma: