Sistemas Operativos II

Comunicación interprocesos UNIX (aspectos prácticos)

Universidad Arturo Jauretche
Ingeniería Informática

Docentes:

Ing. Jorge Osio

Ing. Eduardo Kunysz

Comunicación interprocesos

El SO protege el espacio de memoria de cada proceso

Cada proceso se ejecuta de forma independiente

Si uno falla no afecta el funcionamiento de los demás

Un ejemplo común de fallo podría ser el siguiente:

Variables compartidas

```
int n_hijo=1;
[...]
switch (fork())
   {case -1: /* Error */
    [...]
    case 0:  /* Código del hijo */
        printf("Soy el hijo %d.\n", n_hijo);
        n_hijo=n_hijo+1; /* Para el próximo que tenga mi padre...
*/
        [...]
        exit(0);
    }
/* El padre sigue por aquí y tiene otro hijo... */
switch (fork())
   {case -1: /* Error */
    [...]
    case 0:  /* Código del hijo */
        printf("Soy el hijo %d.\n", n_hijo);
        n_hijo=n_hijo+1; /* Para el próximo que tenga mi padre...
*/
        [...]
        exit(0);
    }
```

Variable compartida

- La variable que modifica el primer hijo n_hijo, ya pertenece a su propio espacio de direcciones. (No modifica a la variable del padre)
- **No es posible** comunicarse entre los dos procesos mediante este procedimiento.
- El padre podría utilizar un **fichero** para comunicarse, pero es un **mecanismo rudimentario y se evita**

Tuberías (Pipe)

- Contribución propia del UNIX a los SS.OO.
- Lo que se escribe en el archivo se almacena en un búffer hasta que alguien lo lea. Se sigue un esquema FIFO.
- Dos tipos de tuberías:
  - Con nombre (named): tienen una entrada de directorio asociada y, por tanto, pueden ser accedidas por cualquier proceso que tenga los permisos adecuados.
  - Sin nombre: no tienen entrada de directorio asociada. Sólo pueden acceder a ella procesos que las hayan heredado en un fork().

Tuberías (Pipe)

- SO permite acceso de un proceso a la vez
- Si el buffer se llena, el proceso que escribe se bloquea.
- Si el buffer está vacío, el proceso que lee se puede bloquear.

Tubería con nombre

- Comando: **mknod** crea la tubería

```
usuario@host:~$ /sbin/mknod tubo p
usuario@host:~$ ls -l
prw-r--r--  1 usuario users      0 abr  8 17:18 tubo
```

Para utilizarla usamos **cat**

En una consola escribimos: **`cat <tubo`**

En otra consola escribimos: **`cat >tubo`**

Lo que se escriba en la segunda consola aparece en la primera (se sale con CTRL+D)

Nota: para usar tuberías en C se utiliza la llamada al sistem mknod

Tubería sin nombre

- Se crean mediante las llamadas a sistema **pipe**

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

Ejemplo de uso:

```
int tubos[2];
[...]
if (pipe(tubos)==-1)
   {/* Error */
    [...]
    }
write(tubo[1],"Hola\n",5);
```

Tuberías sin nombre

- Se declara primero el array de dos enteros.
- El array contiente dos descriptores de ficheros (read y write)
- Se debe abrir mediante la llamada a sistema «open» y cerrar mediante la llamada «close»
- Se pueden utilizar las funciones de string:
  - fprintf
  - fscanf
  - fopen

Sockets

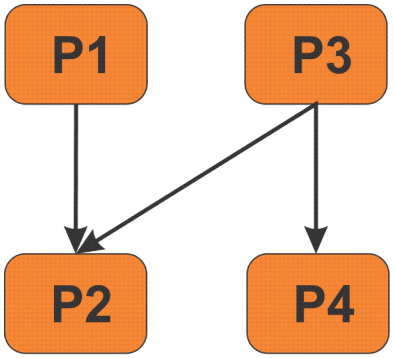
- Objetos del SO parecidos a tuberías.
- Con nombre y sin nombre
- Se utilizan para conectar máquinas distantes via red.

Paso de mensajes

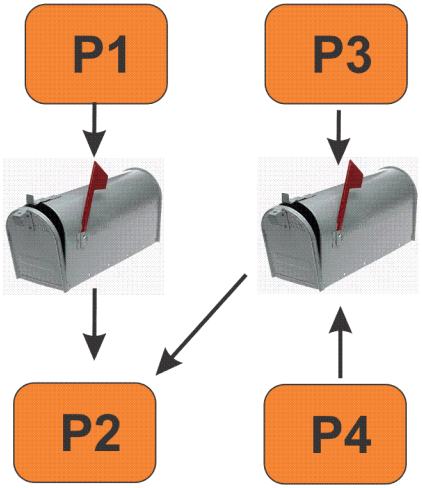
- Herramienta básica de comunicación entre procesos.
- Ventajas:
  - Facil transportar de sistemas monoprocesadores a sistemas de memoria compartida.
  - Fomenta la modularidad y la arquitectura cliente servidor.
- Primitivas:
  - `send(destino, mensaje)`
  - `receive(origen, mensaje)`

Direccionamientos

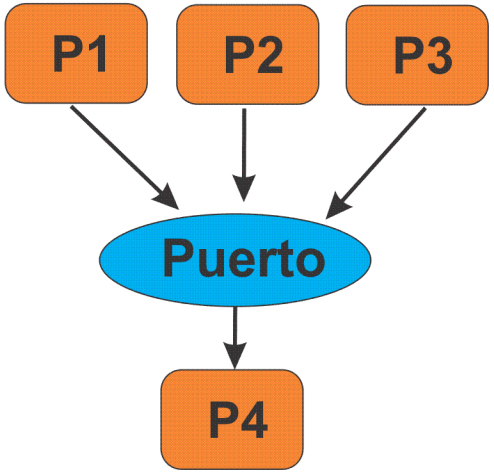
- Directo:
  - Envio: se indica el proceso de destino
  - Recepción: se indica el origen.
- Indirecto: **buzones**



Directo



Indirecto



Indirecto (puertos)

Direccionamiento indirecto

- UNIX utiliza dir. Indirecto (buzones)

```
int msgget(key_t clave, int msgflg);
```

Crea buzones

Tambien conocidos como «queues» (colas)

<u>Formato de los mensajes:</u>

- Nivel de usuario: una estructura con campos para:
  - la longitud y Tipo
  - datos del mensaje.
- Nivel de sistema:
  - Cabecera: origen, destino, longitud del mensaje, información de control, tipo de mensaje.
  - Cuerpo: datos del mensaje.

Ejemplo Buzones

- Se debe definir una estructura del mensaje:

```
#include <sys/msg.h>
#include <stdio.h>

struct tipo_mensaje
   {long tipo; /* Este campo es obligatorio */
    char nombre[20];
    int  edad;};
```

Ejemplo Buzones

- Creo la clave del buzón con ftok

```
int main()
{
 key_t Clave1;
 int Id_Cola_Mensajes;
 Clave1 = ftok ("/bin/ls", 33);
 if (Clave1 == (key_t)-1)
      {
            printf("Error al obtener clave para
                    buzon mensajes");
            exit(-1);
      }
```

Se obtiene una clave a partir de un fichero existente cualquiera y de un entero cualquiera. Todos los procesos que quieran compartir este buzón, deben usar el mismo fichero y el mismo entero.

Ejemplo Buzones

- Creación del buzón

```
Id_Cola_Mensajes = msgget (Clave1, 0600 | IPC_CREAT);
 if (Id_Cola_Mensajes == -1)
     {
          printf( "Error al obtener identificador
                   para cola mensajes");
          exit (-1);
     }
```

0600: hay que darle permisos de acceso al sistema de ficheros UNIX
IPC_CREAT: se debe crear la cola en caso de que no exista
Mediante la orden **ipcs –q** en una consola podemos ver el buzón creado
Se puede eliminar mediante **ipcs rm –q número**

Ejemplo de buzones

- Escribir en el buzon:

```
struct tipo_mensaje m;
[...]
m.tipo=3; /* De tipo 3, por ejemplo */
sprintf(m.nombre,"Pedro");
m.edad=33;
msgsnd(buzOn,&m,sizeof(m)-sizeof(long),0);
```

Manda un mensaje al buzón cuyo identificador es buzOn

Ejemplo Buzones

- Recibir un mensaje

```
int msgrcv(int msqid, void *msgp,
size_t msgsz, long msgtyp, int msgflg);
```

- El primer parámetro es el identificador del buzón, justo como en msgrcv.
- El segundo parámetro es un parámetro de retorno de la función.
- El tercer parámetro es la longitud máxima del mensaje que vamos a leer.

Ejemplo Buzones

- En cuanto al cuarto parámetro, se trata del tipo de mensaje que queremos leer.
  - Si ponemos el tipo 0, recibiremos el primer mensaje que haya en el buzón.
  - Si ponemos un tipo negativo recibiremos el mensaje cuyo tipo sea menor que el valor absoluto especificado
- El quinto parametros es de opciones:
  - 0: la llamada a la función se queda bloqueada hasta que haya un mensaje del tipo indicado
  - IPC_NOWAIT: se vuelve inmediatamente con un error si no hay mensaje de dicho tipo en la cola.

Ejemplo buzones

- Cerrar el buzón

```
msgctl (Id_Cola_Mensajes, IPC_RMID,
        (struct msqid_ds *)NULL);
```