TEMA 2.3. Comunicación entre Procesos. Tuberías

PROFESORES:
Rubén Santiago Montero
Eduardo Huedo Cuesta

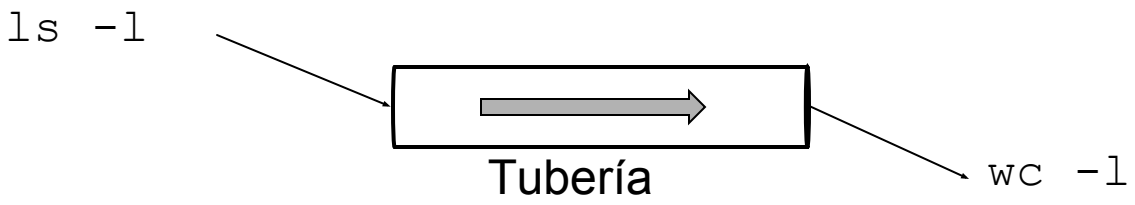
OTROS AUTORES:
Ignacio Martín Llorente
Juan Carlos Fabero Jiménez

Introducción

- Mecanismos de sincronización:
  - Ejecución de procesos/threads en el mismo sistema
    - Señales (sólo para procesos)
    - Ficheros con cerrojos
    - Mutex y variables de condición (solo para threads de un proceso)
    - Semáforos (System V IPC)
    - Colas de mensajes (System V IPC)
  - Ejecución de procesos en distintos sistemas
    - Basados en sockets (paso de mensajes, colas de mensajes…)
- Compartición de datos entre procesos:
  - Ejecución de procesos en el mismo sistema
    - Memoria compartida (System V IPC)
    - Tuberías sin nombre (pipes)
    - Tuberías con nombre (FIFOS)
    - Colas de mensajes (System V IPC)
    - Basados en ficheros
  - Ejecución de procesos en distintos sistemas
    - Basados en sockets

Tuberías sin Nombre

- Soporte para **comunicación unidireccional** entre dos procesos.
- El sistema las **trata** a todos los efectos **como ficheros:**
  - i-nodo
  - Descriptores
  - Tabla de ficheros del sistema y proceso
  - Disponen de las operaciones de E/S típicas
  - Heredadas de padres a hijos
- **Sincronización** realizada por parte del **núcleo**
- Acceso tipo **FIFO** (*first-in-first-out*)
- La tubería **reside** en **memoria principal**



Tuberías sin Nombre

<unistd.h>
SV+BSD+POSIX

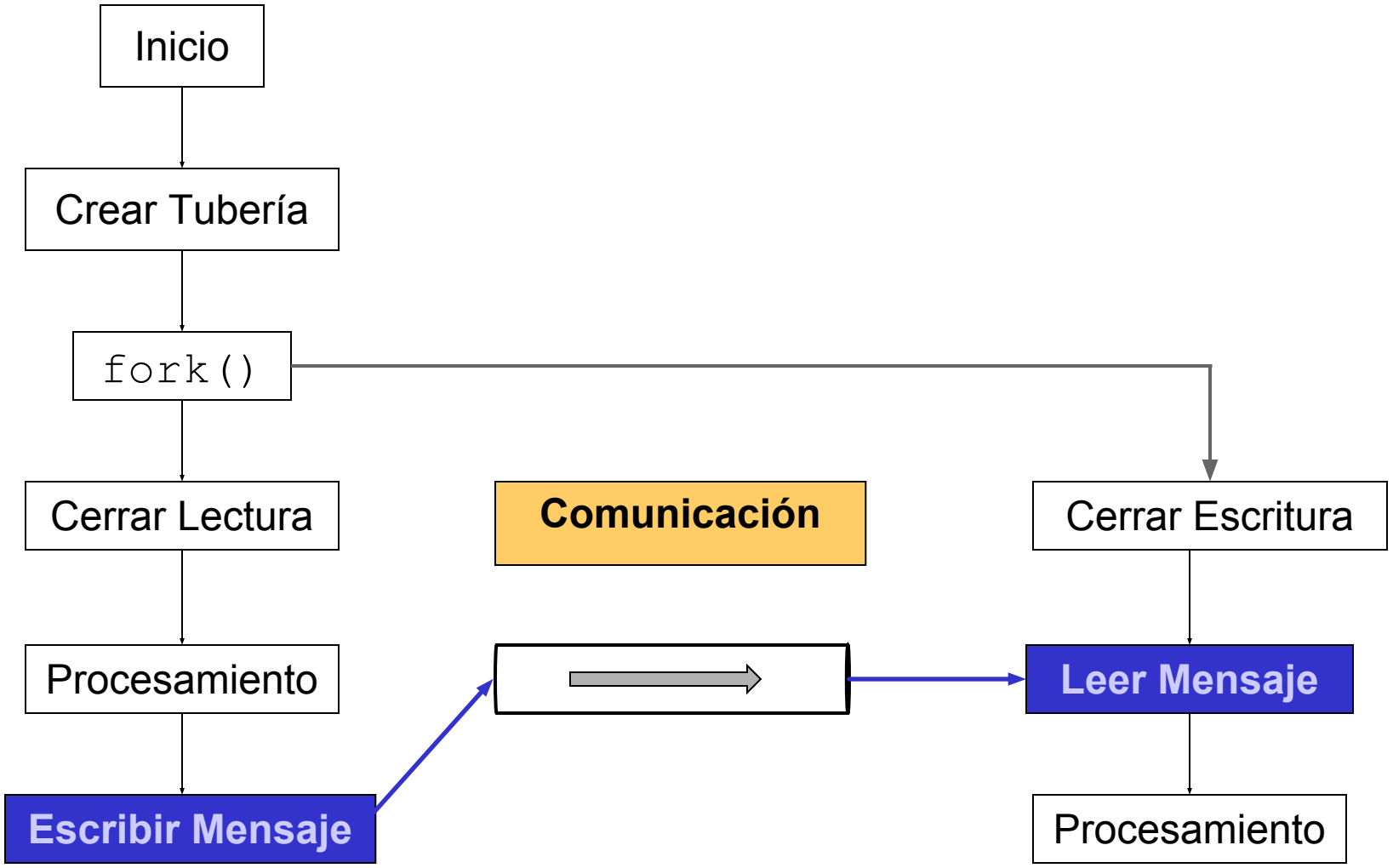
- Creación de una tubería

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

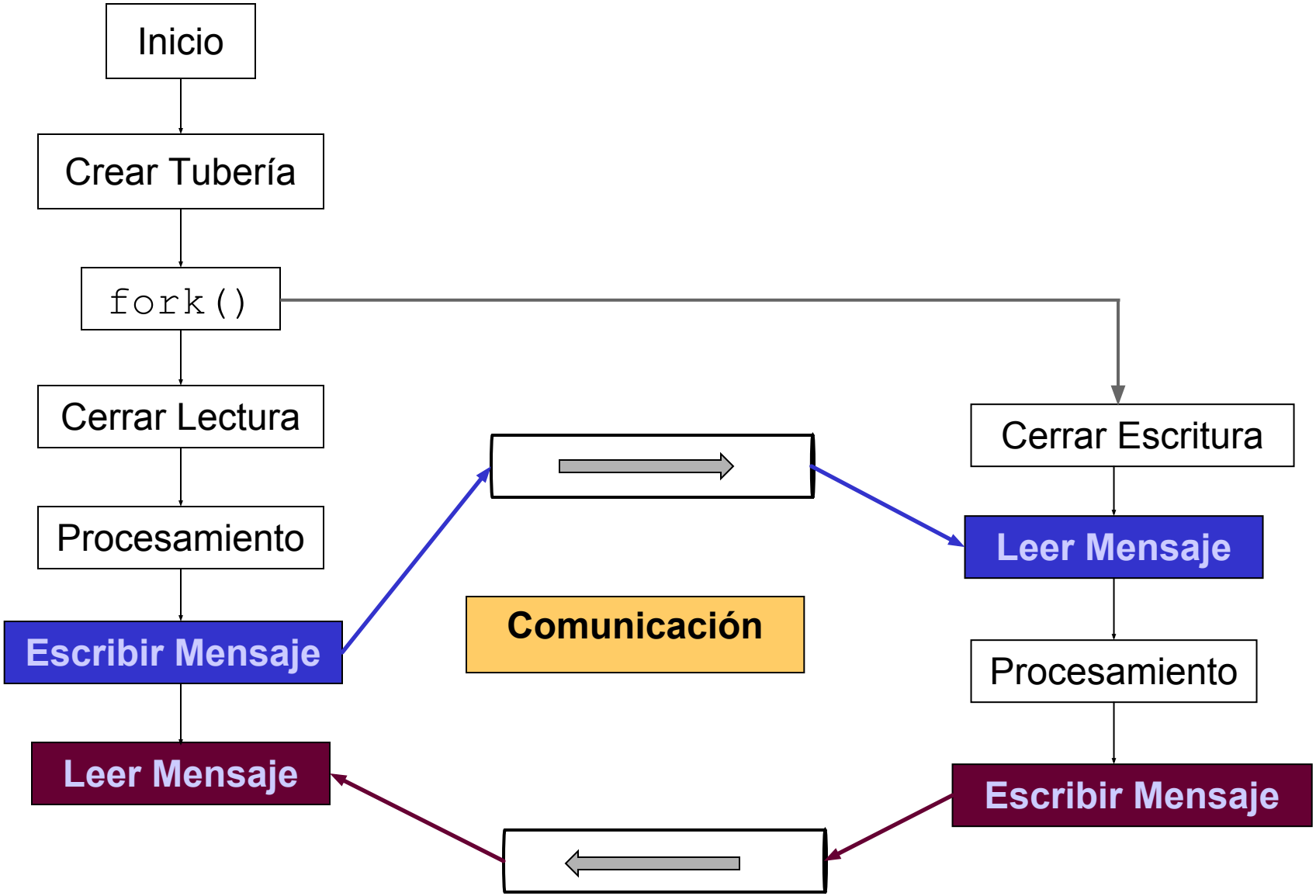
Escritura: `descriptor[1]`  Lectura: `descriptor[0]`

- Si la tubería se llena, las llamadas a `write()` quedarán bloqueadas
- Errores:
  - **EMFILE:** Demasiados descriptores.
  - **ENFILE:** Demasiados ficheros en el sistema.
  - **EFAULT:** Array de descriptores no válido.

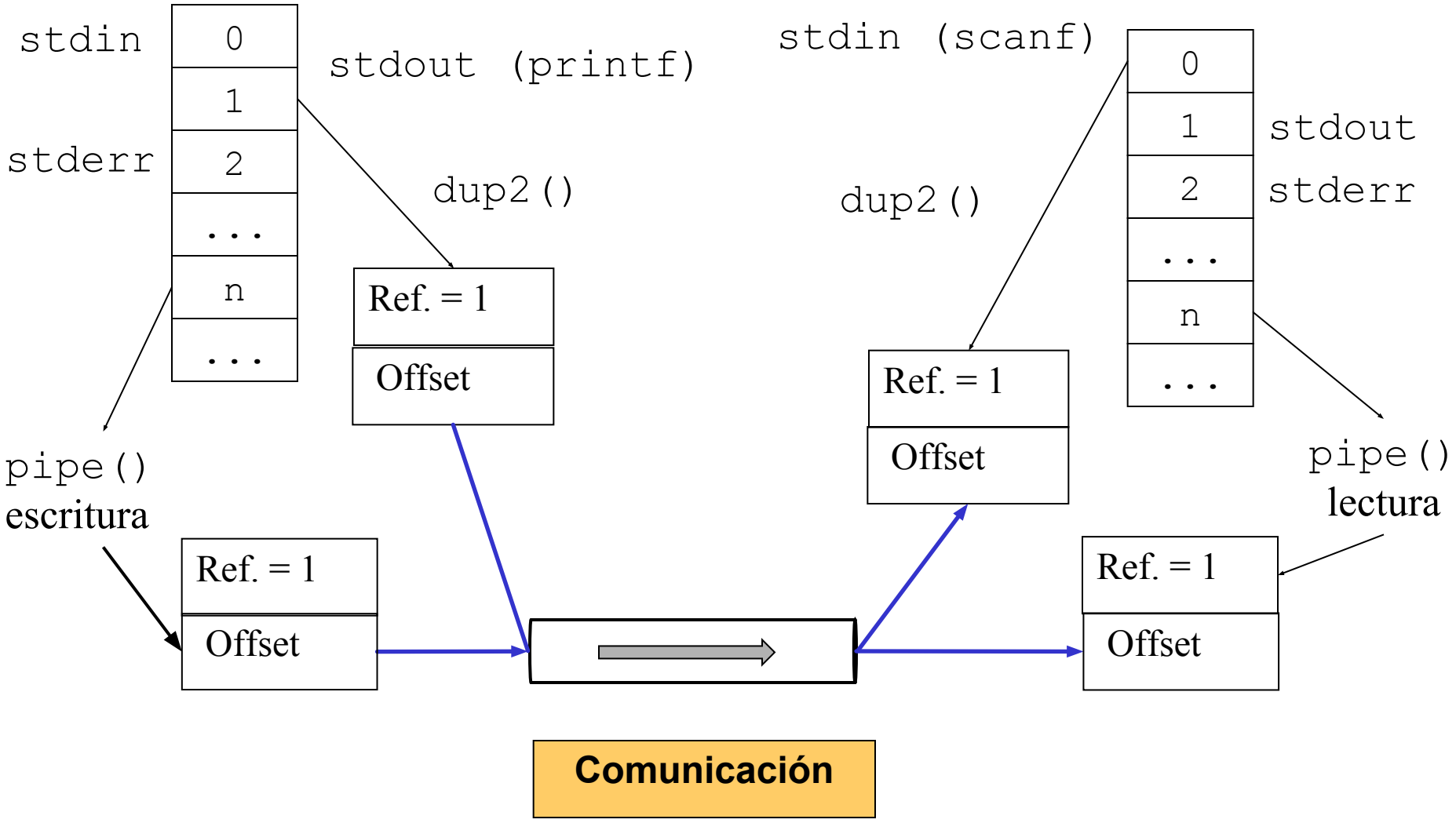
Tuberías sin Nombre



Tuberías sin Nombre



Tuberías sin Nombre



Tuberías con Nombre

- La comunicación mediante **tuberías sin nombre** se realiza únicamente entre **procesos con relación de parentesco**
- **Tubería con nombre**: Es un tipo especial de fichero que ocupa una entrada en el directorio
  - Disponen de la misma funcionalidad que las tuberías sin nombre
  - Las tuberías deben abrirse con la llamada `open()`
  - La sincronización la realiza el núcleo
  - El núcleo almacena los datos internamente, sin escribirlos en el sistema de ficheros

Tuberías con Nombre

- Creación de tuberías (utilidad de la línea de comandos):

```
mknod [-m permisos] nombre tipo
mkfifo [-m permisos] nombre
```

  - `nombre`: Nombre del fichero que se creará
  - `tipo`: El tipo del archivo puede ser
    - `b`: archivo por bloques
    - `c`: archivo por caracteres
    - **`p`: FIFO**

- Creación de tuberías (API sistema):

`<sys/types.h>`
`<sys/stat.h>`
`<fcntl.h>`
`<unistd.h>`
SV+BSD

```
int mknod(const char *filename, mode_t mode, dev_t dev);
```

  - `filename`: Nombre del fichero (archivo, dispositivo, tubería) que se creará
  - `mode`: Especifica los permisos y el tipo de archivo que se creará. Su valor se fijará mediante OR lógica de permisos (umask). El tipo ha de ser:
    - `S_IFREG`: Archivo regular
    - `S_IFCHR`: Archivo de caracteres (dev = major,minor)
    - `S_IFBLK`: Archivo por bloques (dev = major,minor)
    - `S_IFIFO`: Tubería con nombre

Tuberías con Nombre

<sys/types.h>
<sys/stat.h>

POSIX

- Creación de un archivo de tubería (API del sistema):

```
int mkfifo(const char *filename, mode_t mode);
```

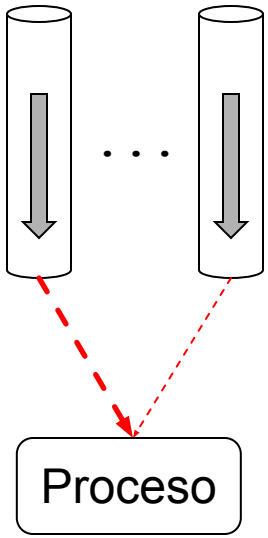
  - `filename`: Nombre de la tubería que se creará
  - `mode`: Determina los permisos con que se creará la tubería. Estos permisos se ven modificados por el umask del proceso

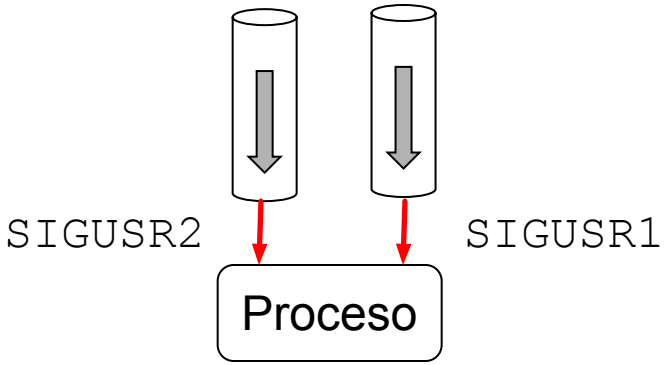
- La apertura de tuberías bloquea el proceso hasta que se conecte un proceso al otro lado de la tubería, este comportamiento puede modificarse con el flag `O_NONBLOCK`

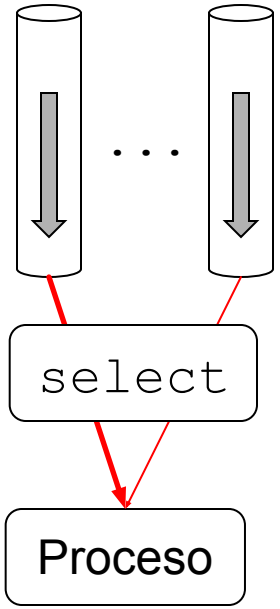
Operación E/S	Situación	Resultado
Lectura	FIFO vacía, con escritor	Se bloquea
Lectura	FIFO vacía, sin escritor	Devuelve 0
Escritura	FIFO llena, con lector	Se bloquea
Escritura	No hay lector conectado	Recibe `SIGPIPE`

Sincronización de E/S

- Cuando un proceso gestiona varios canales de E/S, debe seleccionar los que están listos en cada momento para realizar la operación
- Alternativas:
  - E/S no bloqueante (encuesta)
  - E/S conducida por eventos
  - Multiplexación de E/S síncrona







Multiplexación de E/S Síncrona

<sys/select.h>
POSIX+BSD

- Selección en cada momento del descriptor de fichero que esté listo para realizar la operación de entrada/salida, permitiendo realizarla de forma **síncrona**
- Selección del canal:

```
int select(int n, fd_set *Rset, fd_set *Wset,
           fd_set *Eset, struct timeval *tout);
```

  - `n`: El mayor de los descriptores en cualquiera de los tres conjuntos, más 1
  - `Rset`: Conjunto de descriptores de lectura, se comprobará si hay datos disponibles
  - `Wset`: Conjunto de descriptores de escritura, se comprobará si es posible escribir de forma inmediata
  - `Eset`: Conjunto de descriptores de excepción, se comprobará si hay alguna condición especial
  - `tout`: Tiempo máximo en el que retornará la función. Si es 0, retorna inmediatamente. Si es NULL, se bloquea hasta que se produce un cambio

Multiplexación de E/S Síncrona

- El conjunto de descriptores en los que se ha producido algún cambio de condición se almacena en las variables `Wset`, `Rset` y `Eset` según corresponda
- Macros para la manipulación de los conjuntos:
  - `FD_ZERO(fd_set *set)`: Inicializa `set` como conjunto vacío
  - `FD_SET(int fd, fd_set *set)`: Añade `fd` a `set`
  - `FD_CLEAR(int fd, fd_set *set)`: Elimina `fd` de `set`
  - `FD_ISSET(int fd, fd_set *set)`: Comprueba si `fd` está en `set`
- La función devuelve el número de descriptores que han experimentado un cambio de estado
- Si se produce un error, los conjuntos no se modifican y `tout` queda indeterminado
  - **`EBADF:`** Descriptor no válido en alguno de los tres conjuntos
  - **`EINTR:`** Señal no bloqueada recibida
  - **`EINVAL:`** Valor de n negativo

Multiplexación de E/S Síncrona

```
...
FD_ZERO(&conjunto);
FD_SET(0, &conjunto);

timeout.tv_sec = 2;
timeout.tv_usec = 0;

cambios = select(1, &conjunto, NULL, NULL, &timeout);
if (cambios == -1)
    perror("select()");
else if (cambios) {
    printf("Datos nuevos.\n");
    scanf("%s", buffer);
    printf("Datos: %s\n", buffer);
} else {
    printf("Ningún dato nuevo en 2 seg.\n");
}
...
```