Sistemas Operativos

sesión 12: tuberías

Grado en Ingeniería Informática

Universidad Carlos III de Madrid

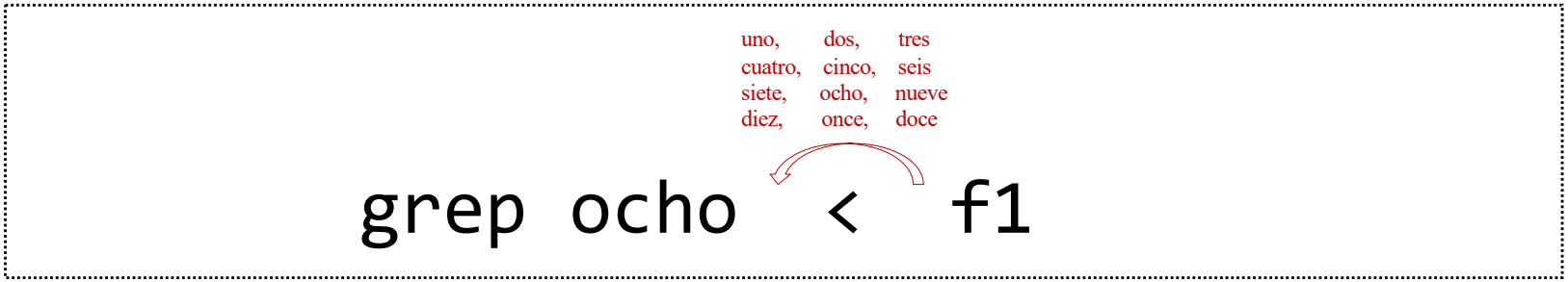
Contenidos

- **Redirección**
- Tuberías (pipes)

Ejemplo de redirección de entrada

f1.txt

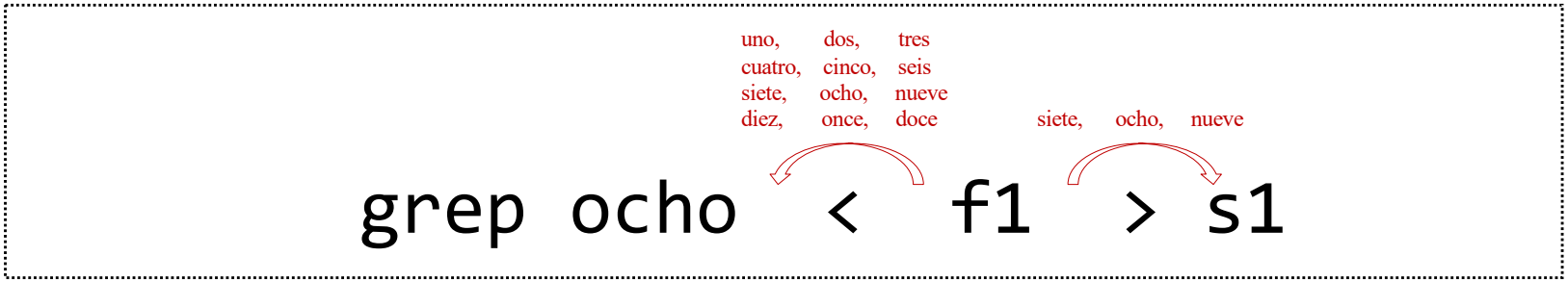
uno,	dos,	tres
cuatro,	cinco,	seis
siete,	ocho,	nueve
diez,	once,	doce



Ejemplo de redirección de salida

f1.txt

uno,	dos,	tres
cuatro,	cinco,	seis
siete,	ocho,	nueve
diez,	once,	doce



Ejemplo de redirección de salida

f1.txt

uno,	dos,	tres
cuatro,	cinco,	seis
siete,	ocho,	nueve
diez,	once,	doce

Dependiente del intérprete de mandatos usado


```
grep ocho f1  1>  s1
```

Ejemplo de redirección de error

f1.txt

uno,	dos,	tres
cuatro,	cinco,	seis
siete,	ocho,	nueve
diez,	once,	doce

Dependiente del intérprete de mandatos usado

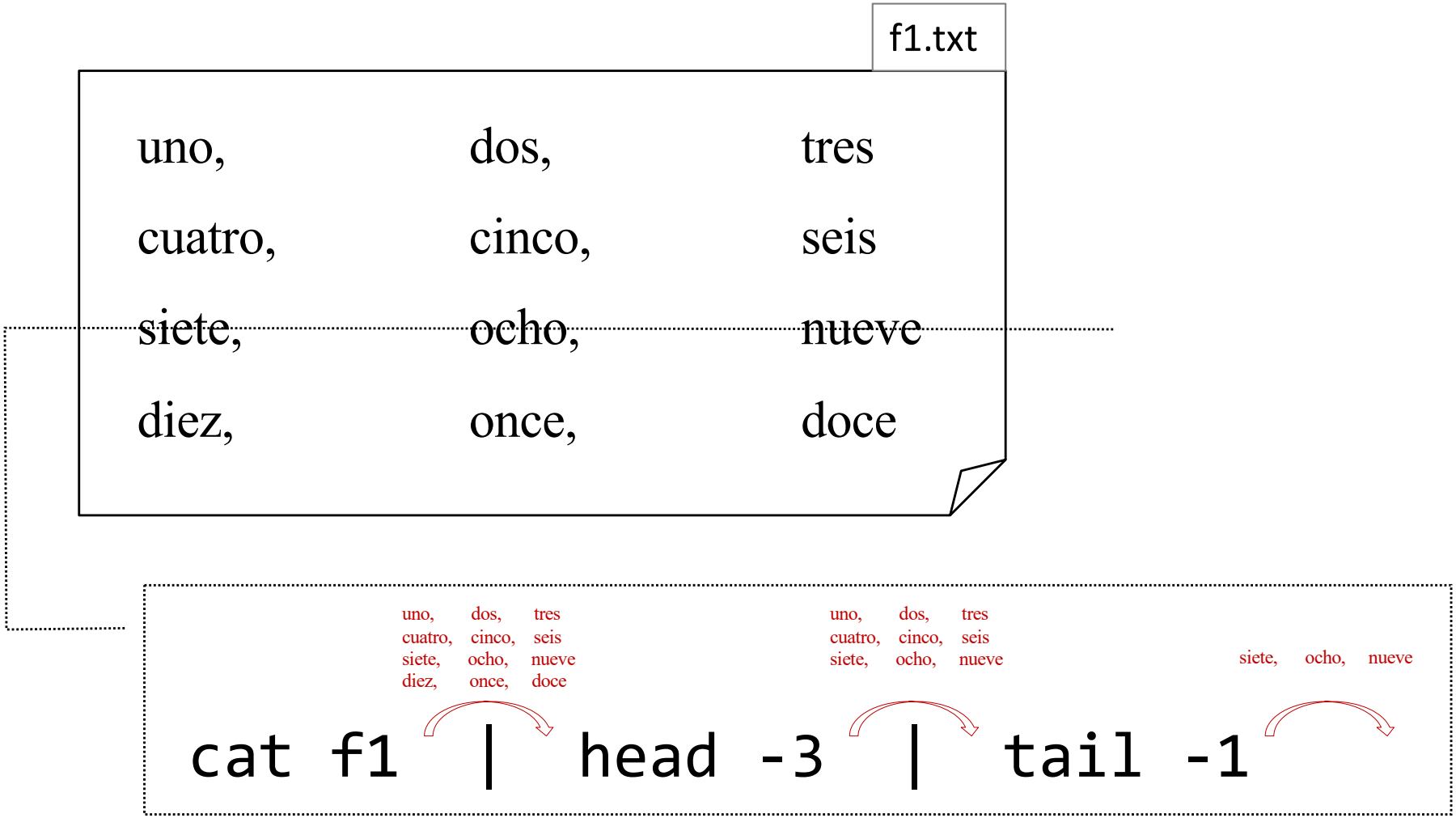
```
grep ocho xx 2> s1
```

grep: f2: No existe el archivo o el directorio

Contenidos

- Redirección
- **Tuberías (pipes)**

Ejemplo de uso de tuberías



Contenidos

- **Los descriptores de ficheros**
  - Redirección y duplicado
- Los descriptores de ficheros y *fork*()
- Tuberías

Descriptores de ficheros

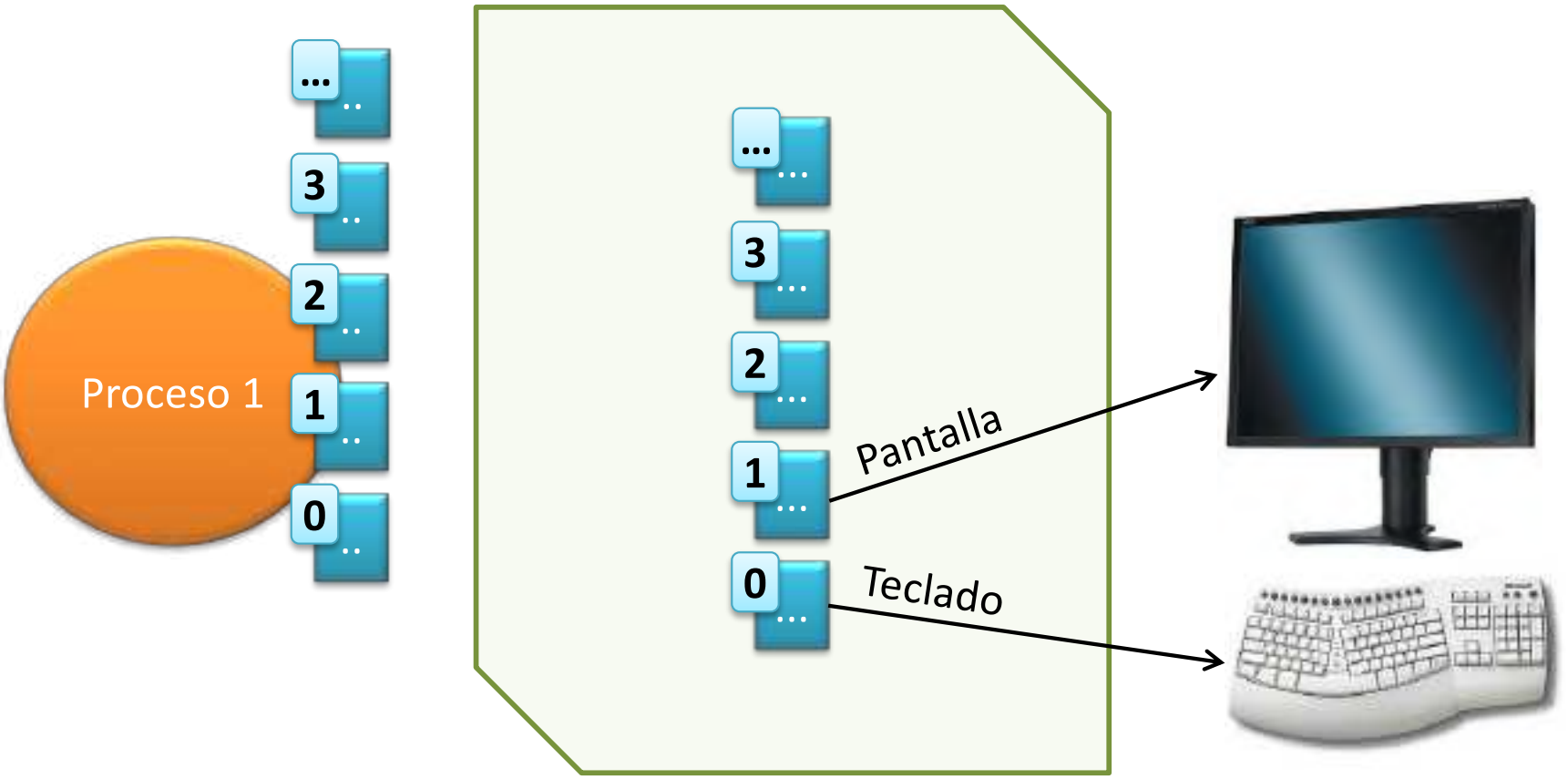
Los descriptores de ficheros son el índice de la tabla que hay por proceso
que identifica los posibles ficheros (o dispositivos) con los que comunicarse

Descriptores de ficheros

Por defecto se utilizan los tres primeros para la entrada estándar, salida estándar y salida de error respectivamente.

Descriptores de ficheros

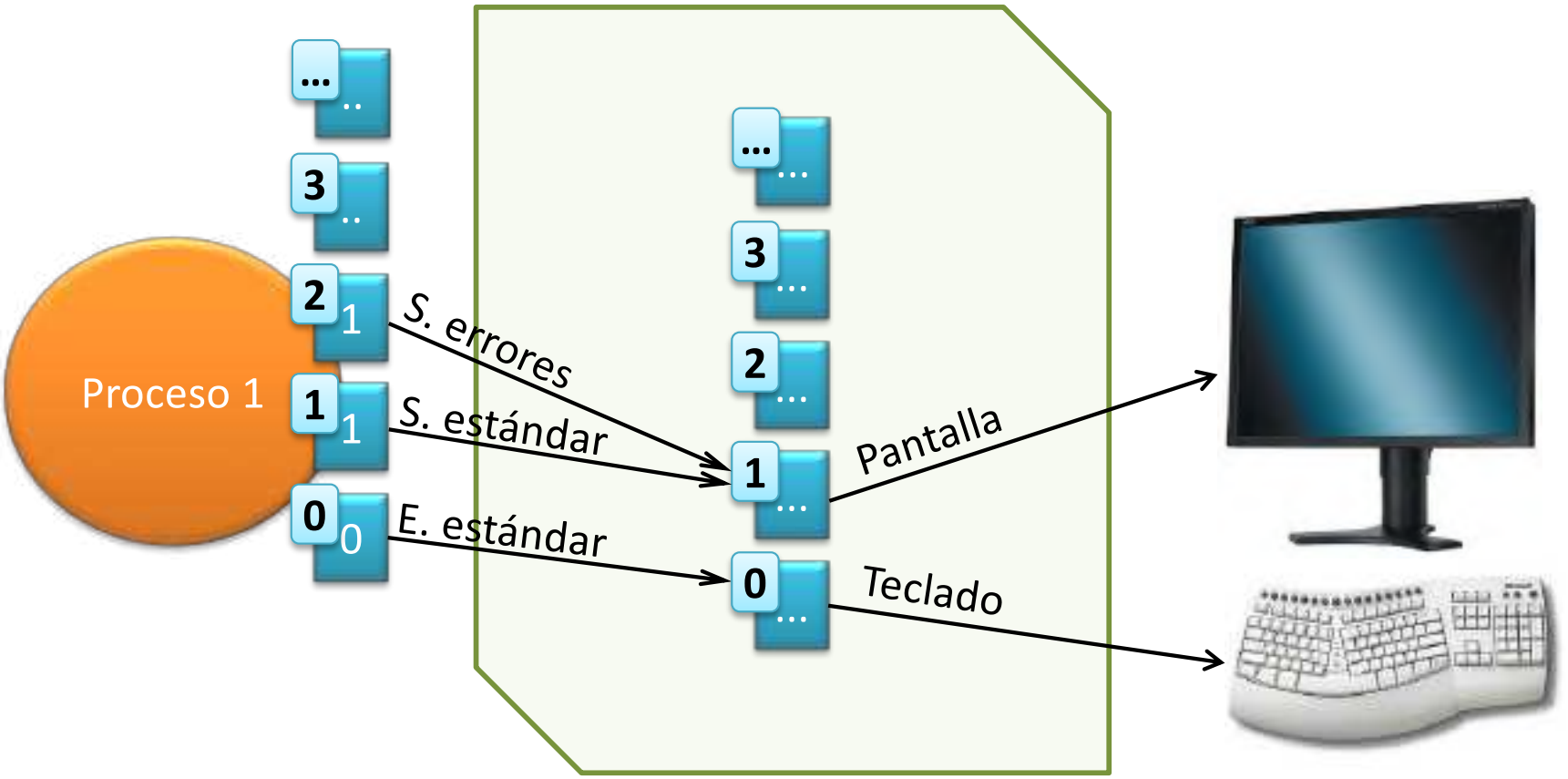
abstracción ofrecida



El sistema operativo mantiene una tabla interna con la información real de contacto con los dispositivos y ficheros con los que los procesos piden comunicarse…

Descriptores de ficheros

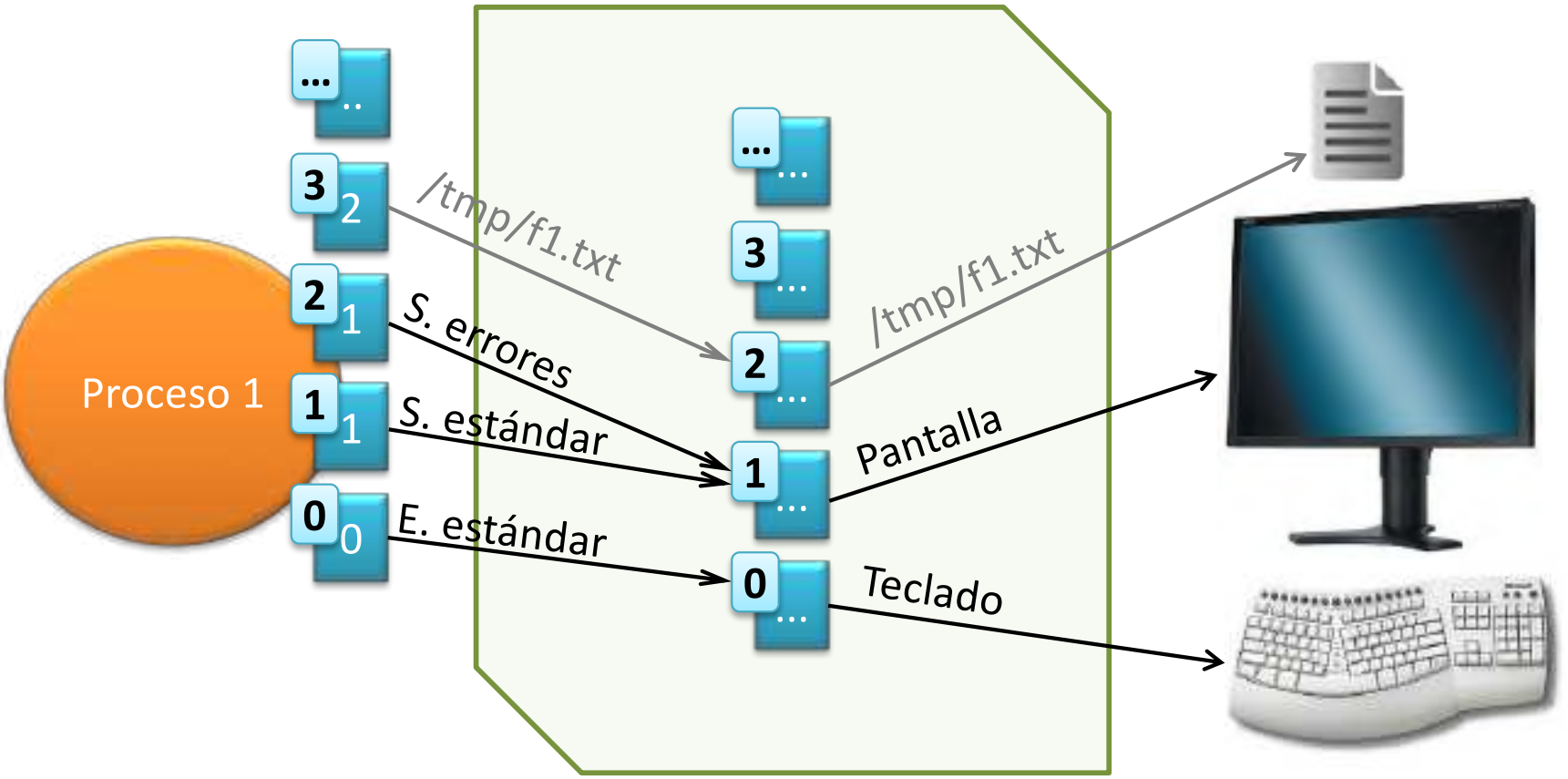
abstracción ofrecida



…Y los descriptores de ficheros son el índice de la tabla que hay por proceso, cuyo contenido es a su vez el índice de la tabla interna del sistema operativo.

Descriptores de ficheros

abstracción ofrecida



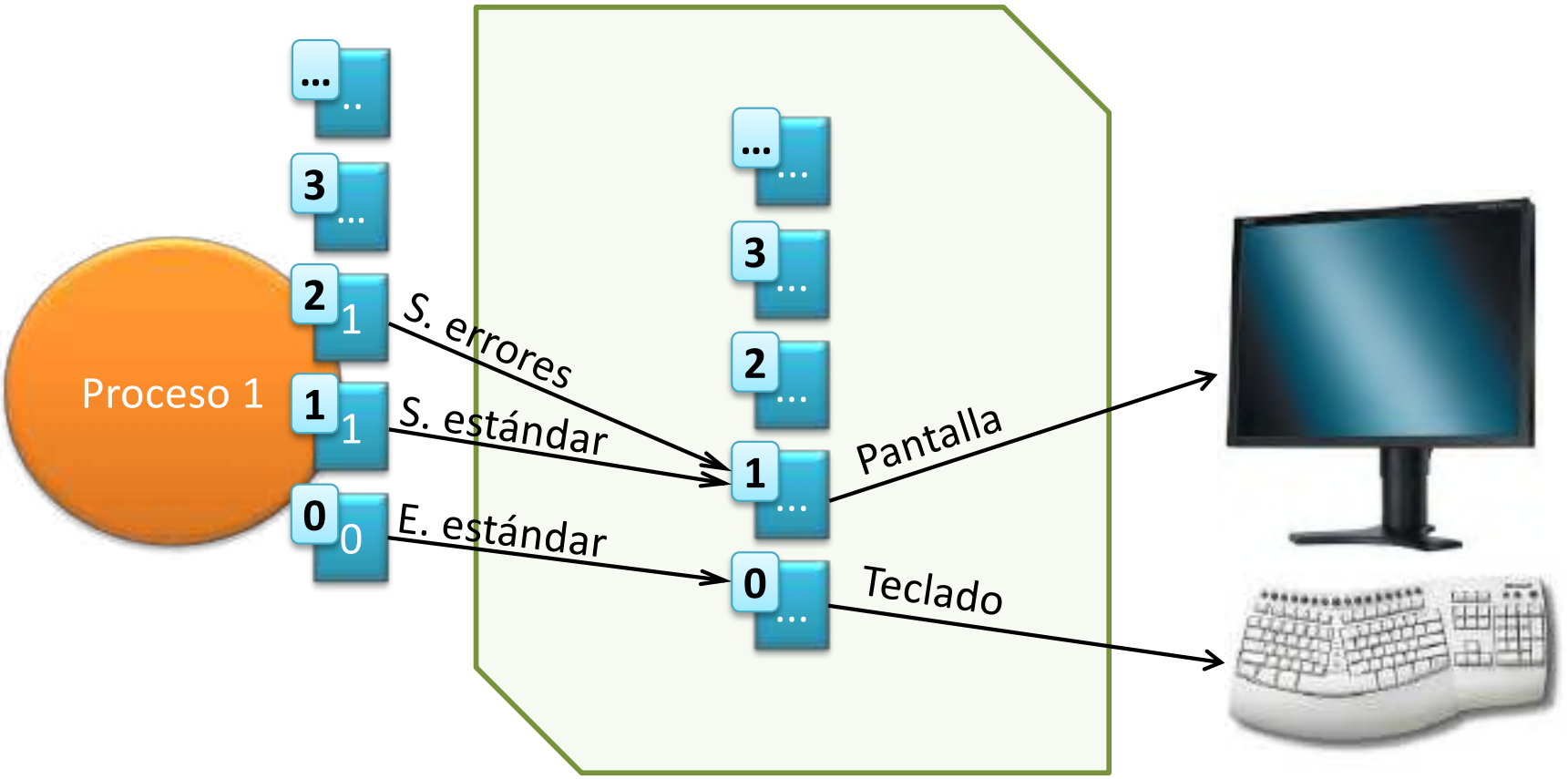
Cuando se pide un nuevo descriptor de ficheros (al abrir un fichero) se busca el primero hueco libre de la tabla y el índice de esa posición es el descriptor asignado.

Contenidos

- Los descriptores de ficheros
  - **Redirección y duplicado**
- Los descriptores de ficheros y *fork*()
- Tuberías

Descriptores de ficheros

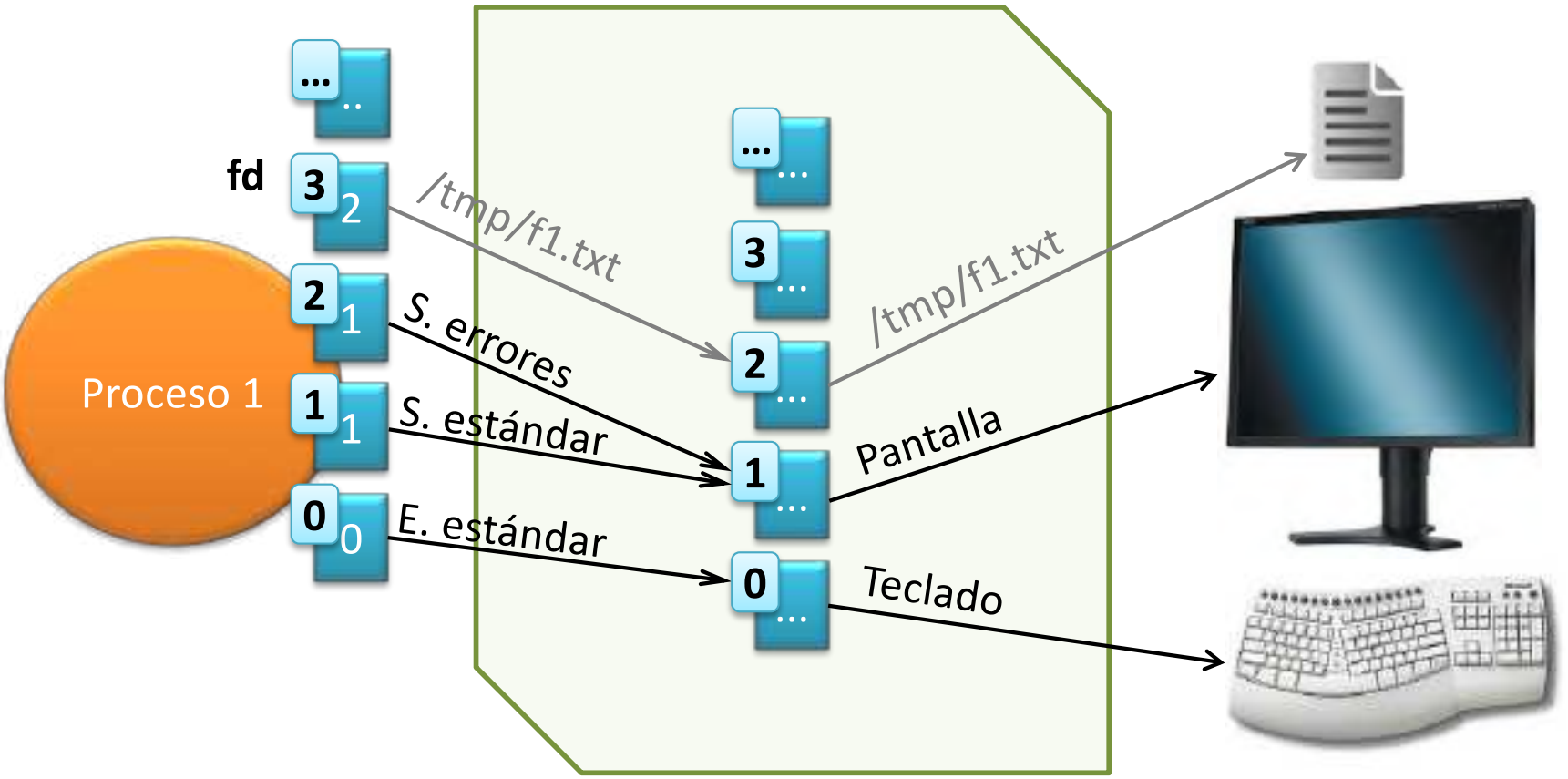
redirección a fichero




```
> /tmp/f1.txt
```

Descriptores de ficheros

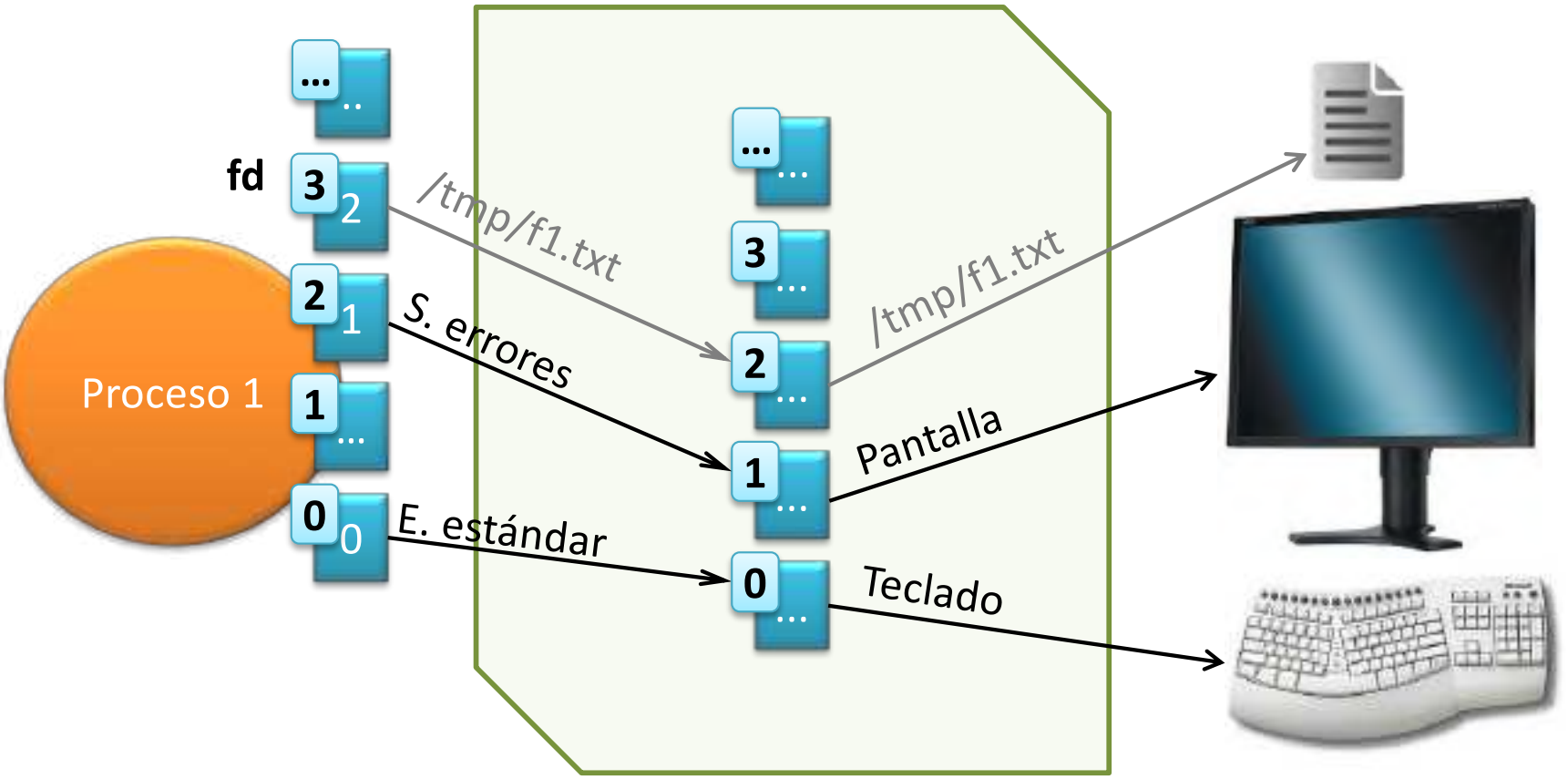
redirección a fichero




```
fd = creat ("/tmp/f1.txt", 0644);
```

Descriptores de ficheros

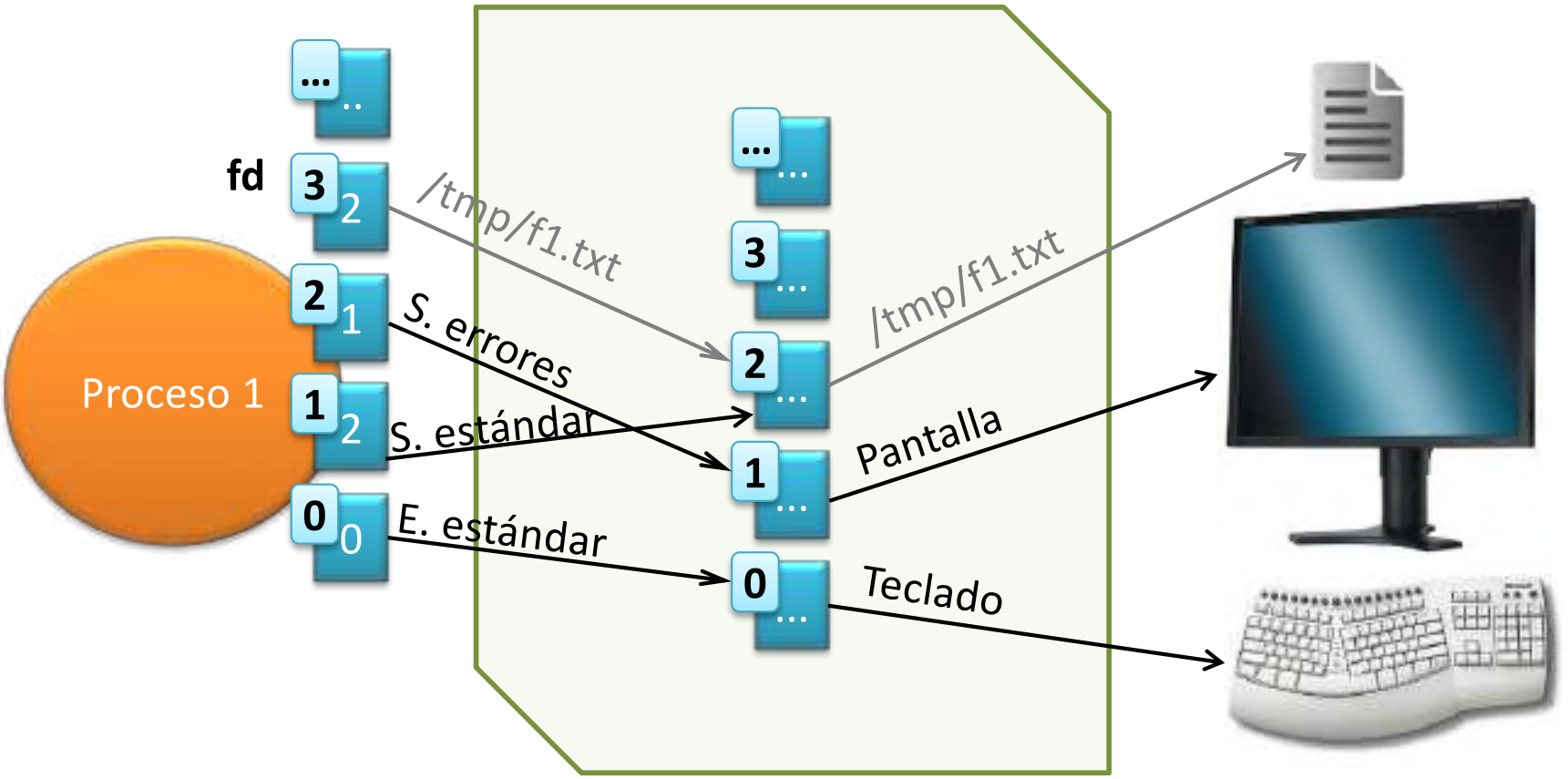
redirección a fichero



close (1)

Descriptores de ficheros

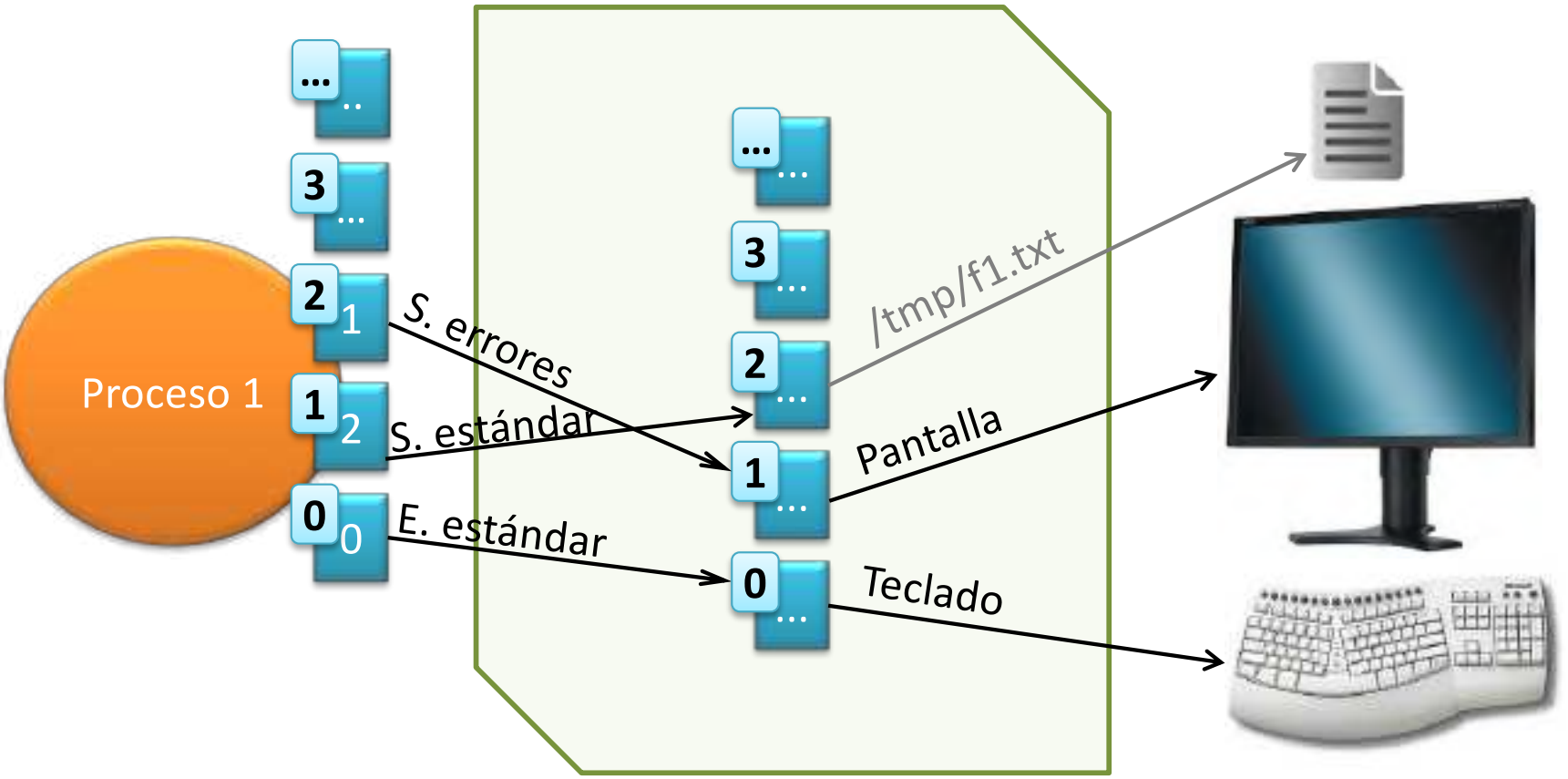
redirección a fichero



dup (fd)

Descriptores de ficheros

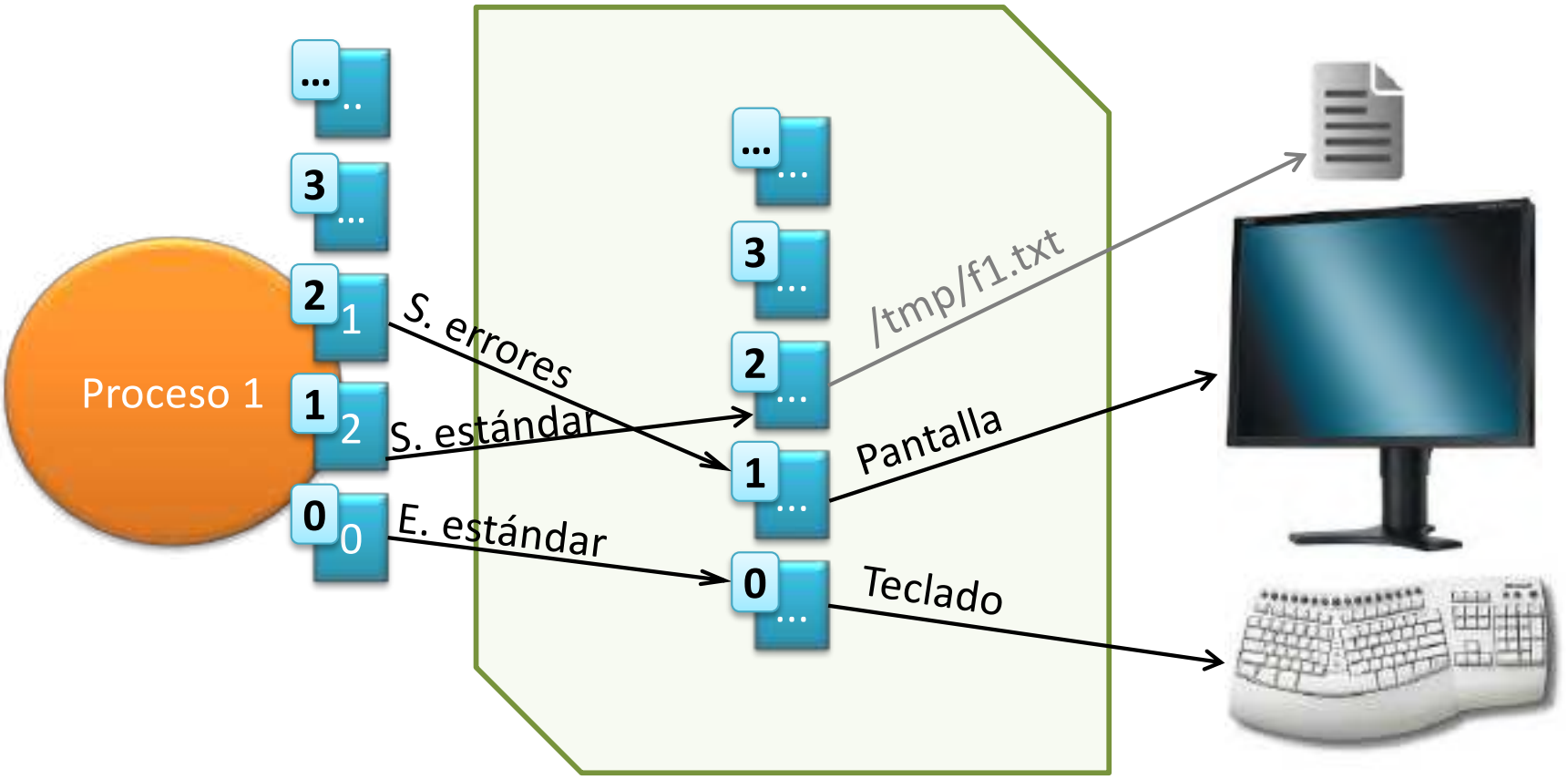
redirección a fichero



close(fd)

Descriptores de ficheros

redirección a fichero



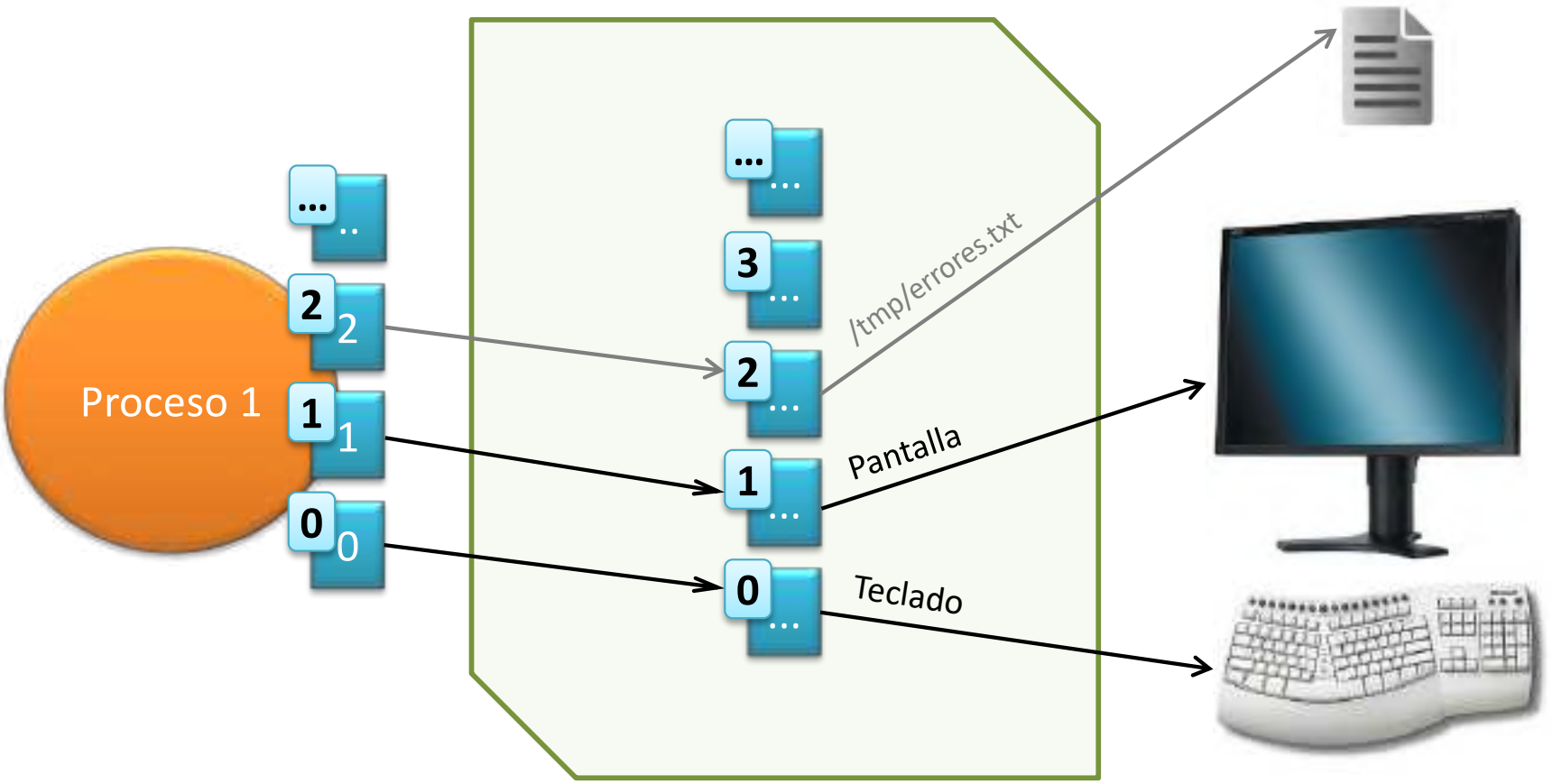
La salida estándar ha quedado redirigida al fichero /tmp/f1.txt

Contenidos

- Los descriptores de ficheros
  - Redirección y duplicado
- **Los descriptores de ficheros y *fork*()**
- Tuberías

Descriptores de ficheros

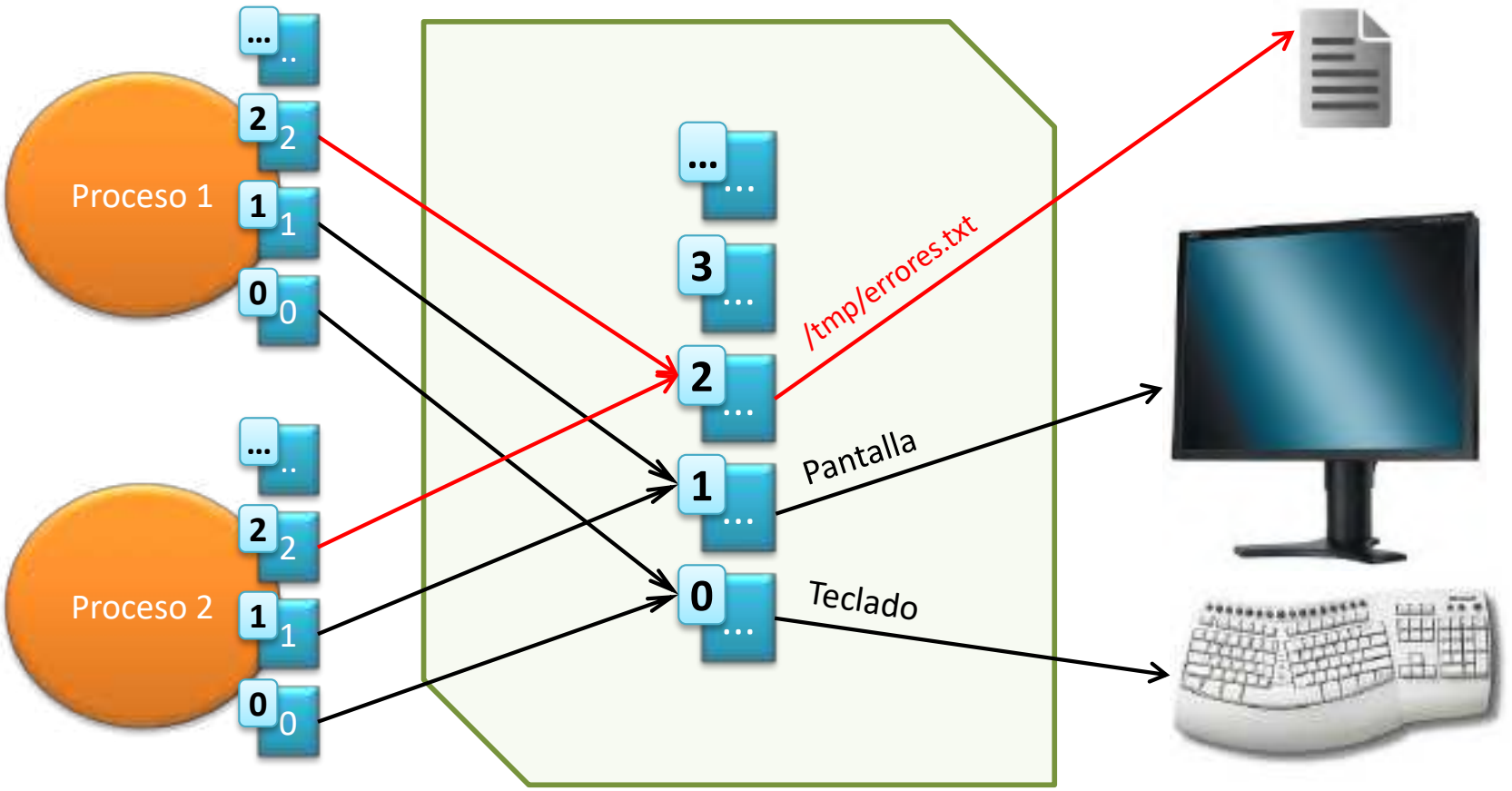
llamada fork()



fork() crea un duplicado del hijo

Descriptores de ficheros

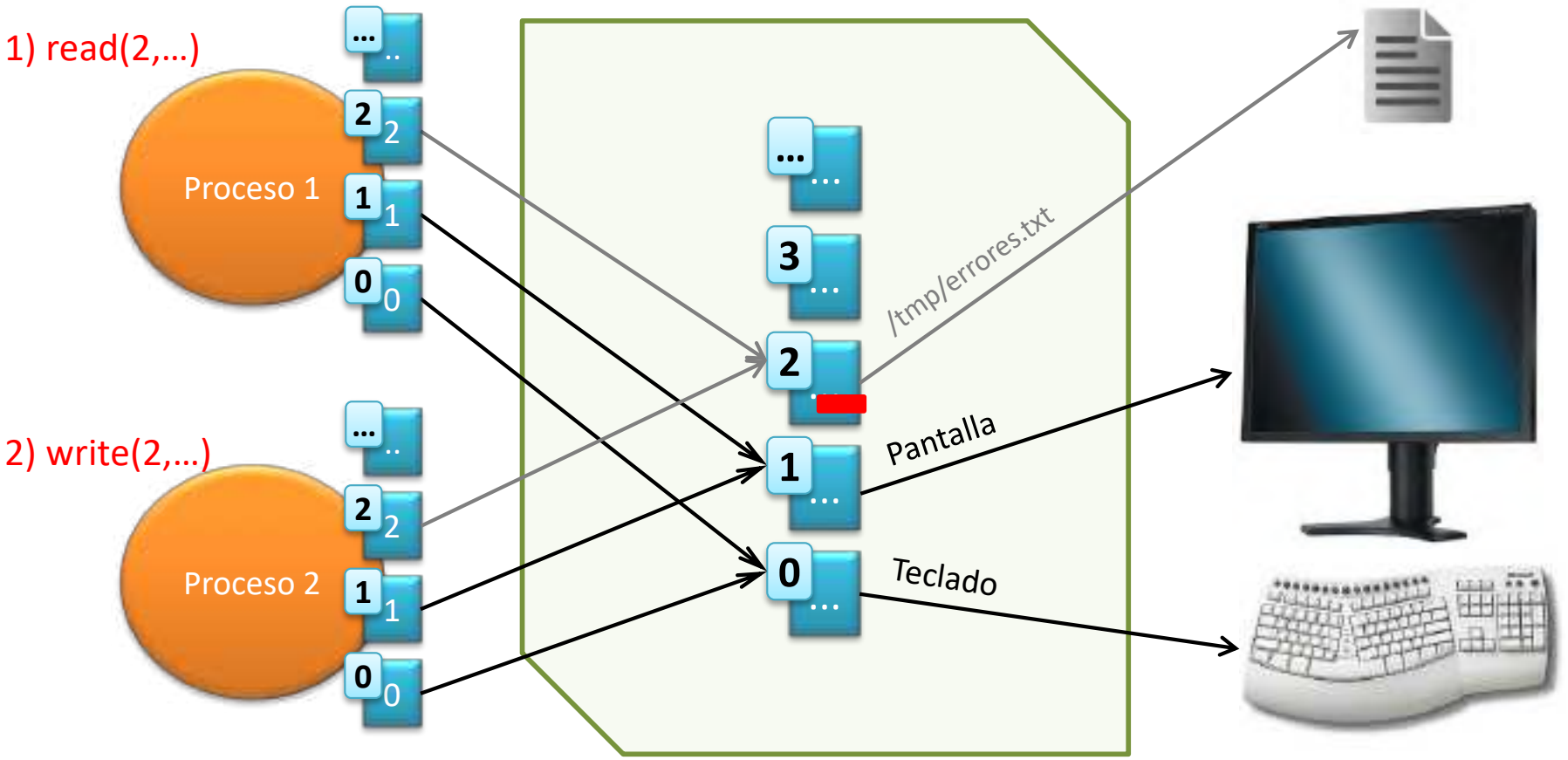
llamada fork()



- **Ambos tienen descriptores iguales (redirecciones antes del *fork()* se heredan)**
- Ambos referencian los mismos elementos (posición L/E después del *fork()* común)

Descriptores de ficheros

llamada fork()



- Ambos tienen descriptores iguales (redirecciones antes del *fork()* se heredan)
- **Ambos referencian los mismos elementos (posición L/E después del *fork()* común)**

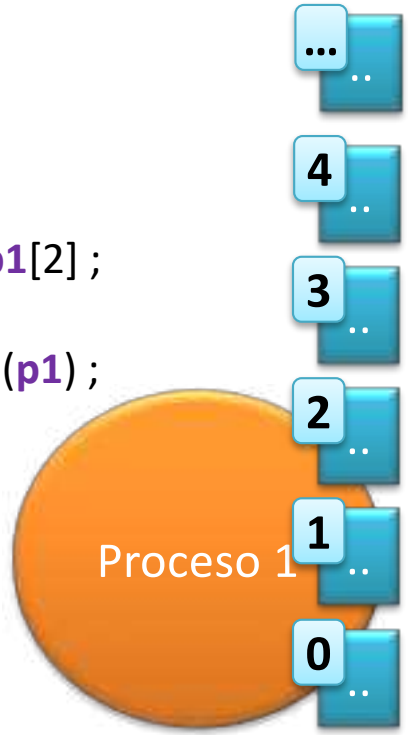
Contenidos

- Los descriptores de ficheros
  - Redirección y duplicado
- Los descriptores de ficheros y *fork*()
- **Tuberías**

Tubería

1 creación

```
int p1[2] ;
…
pipe(p1) ;
…
```

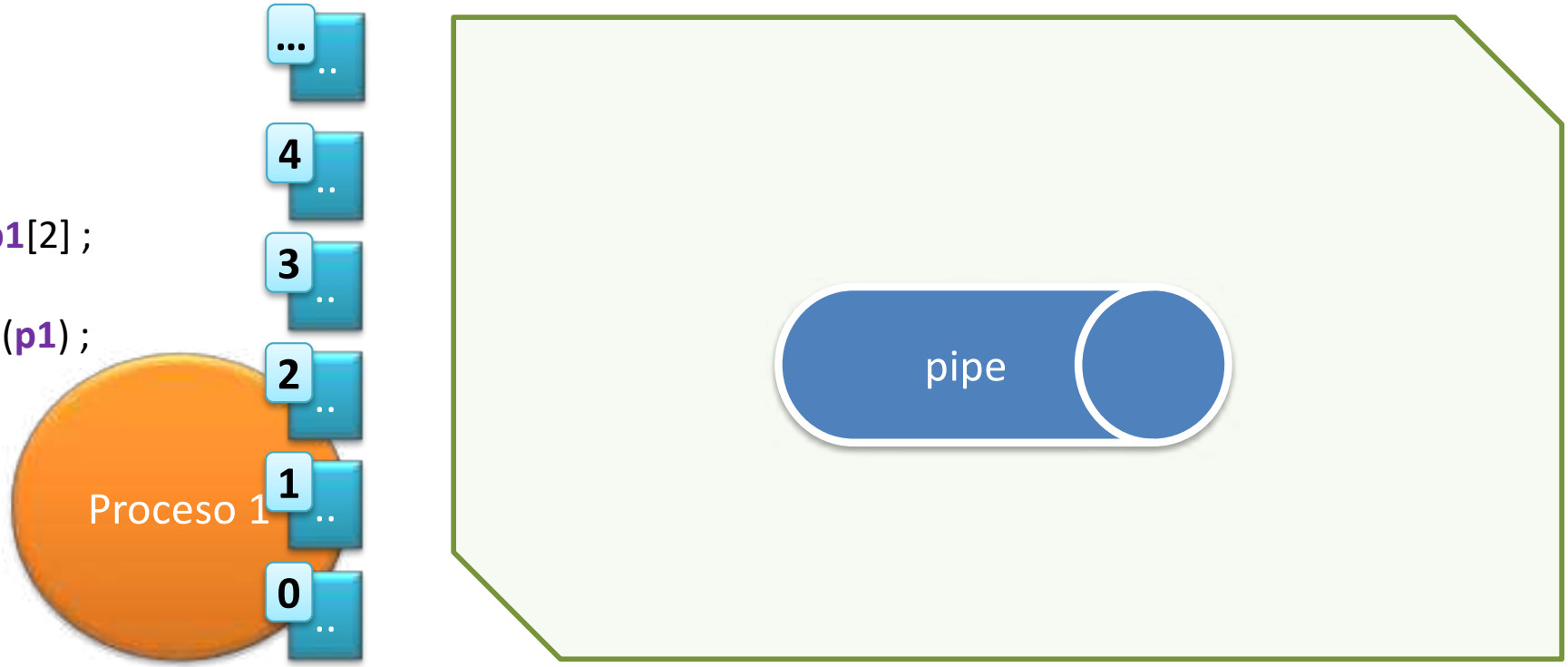


Una tubería es un fichero especial que se crea con la llamada al sistema *pipe()*

Tubería

1 creación

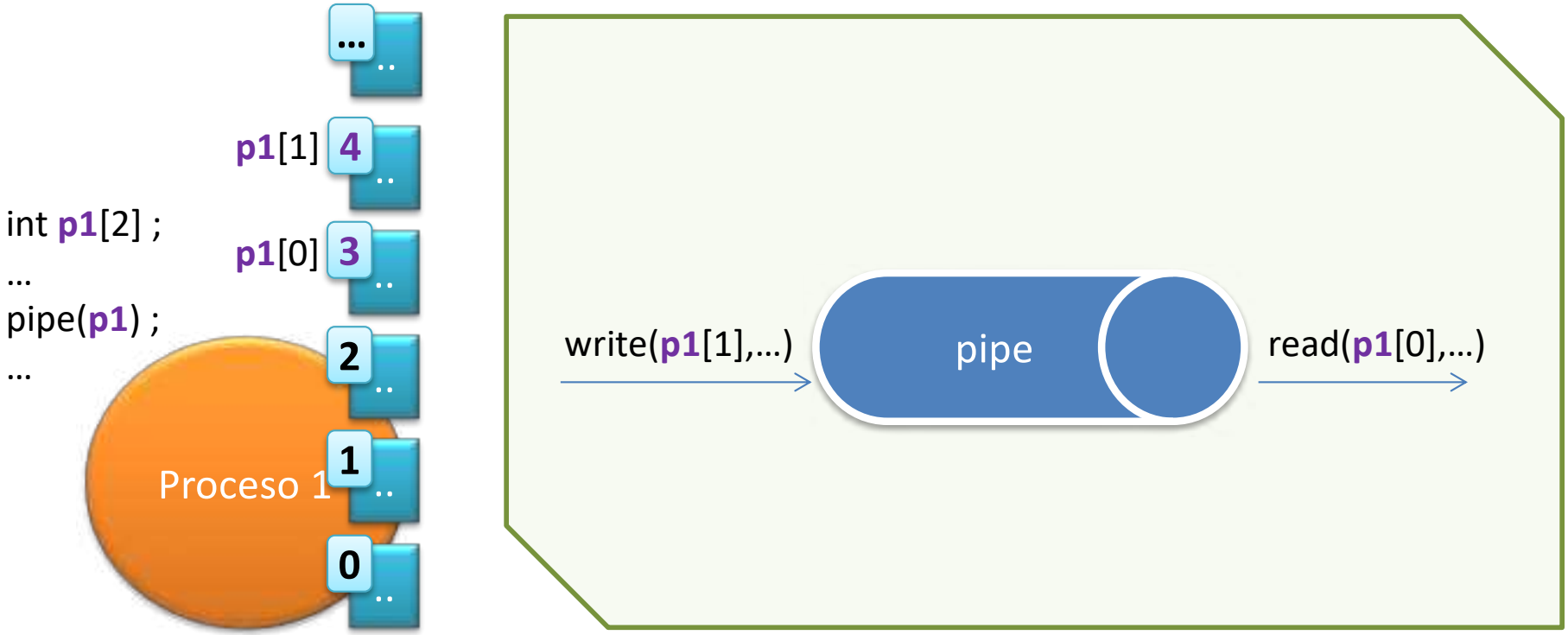
```
int p1[2] ;
…
pipe(p1) ;
…
```



Una tubería es un fichero especial que se crea con la llamada al sistema *pipe()*

Tubería

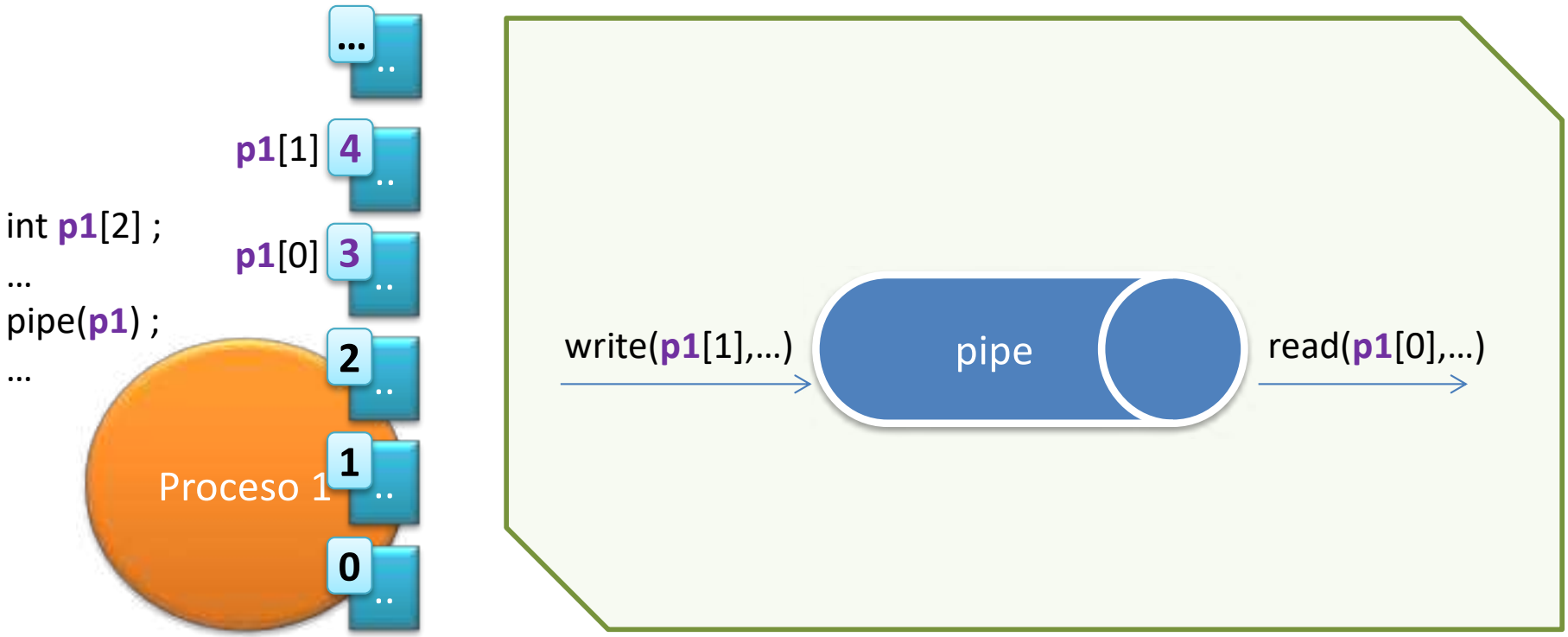
1 creación



Una tubería es un fichero especial que se crea con la llamada al sistema *pipe()*
Dicha llamada crea la tubería y reserva dos descriptores de ficheros: lectura y escritura

Tubería

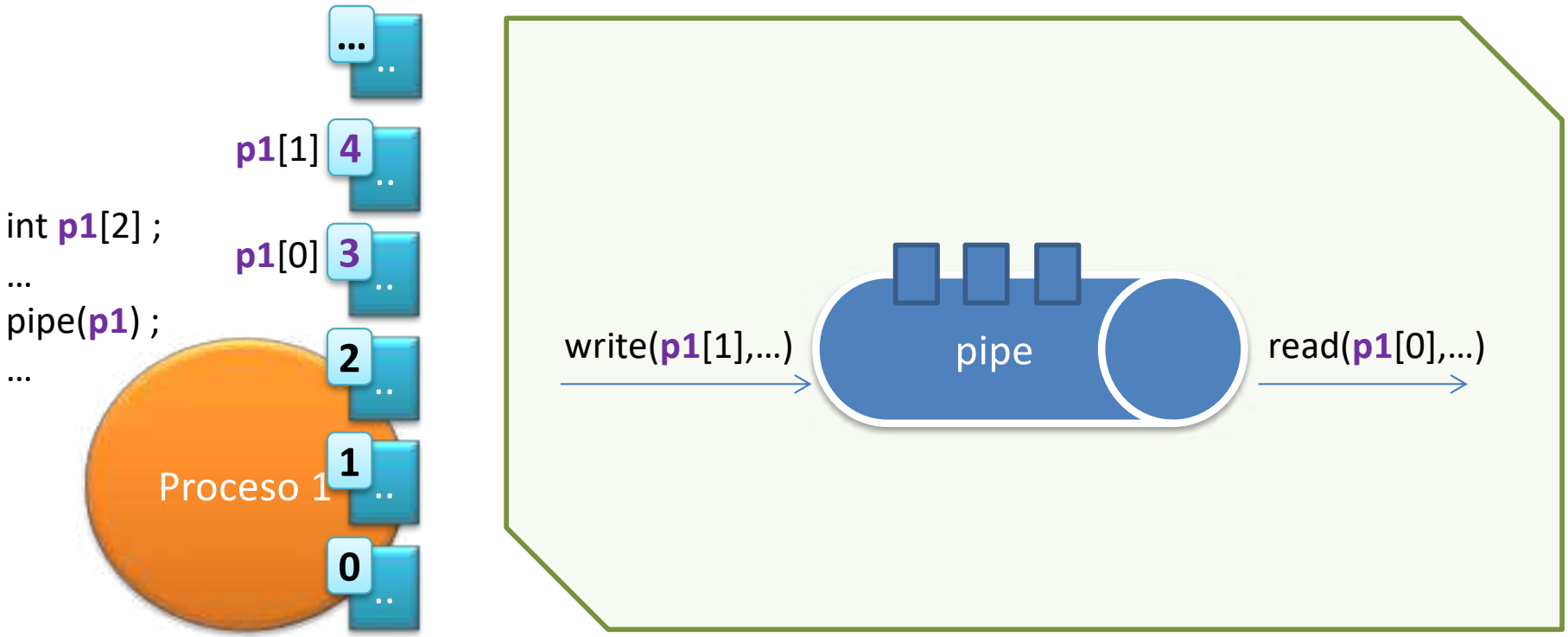
1 creación



Una operación de escritura inserta datos en el pipe
Una operación de lectura extrae datos del pipe

Tubería

1 creación

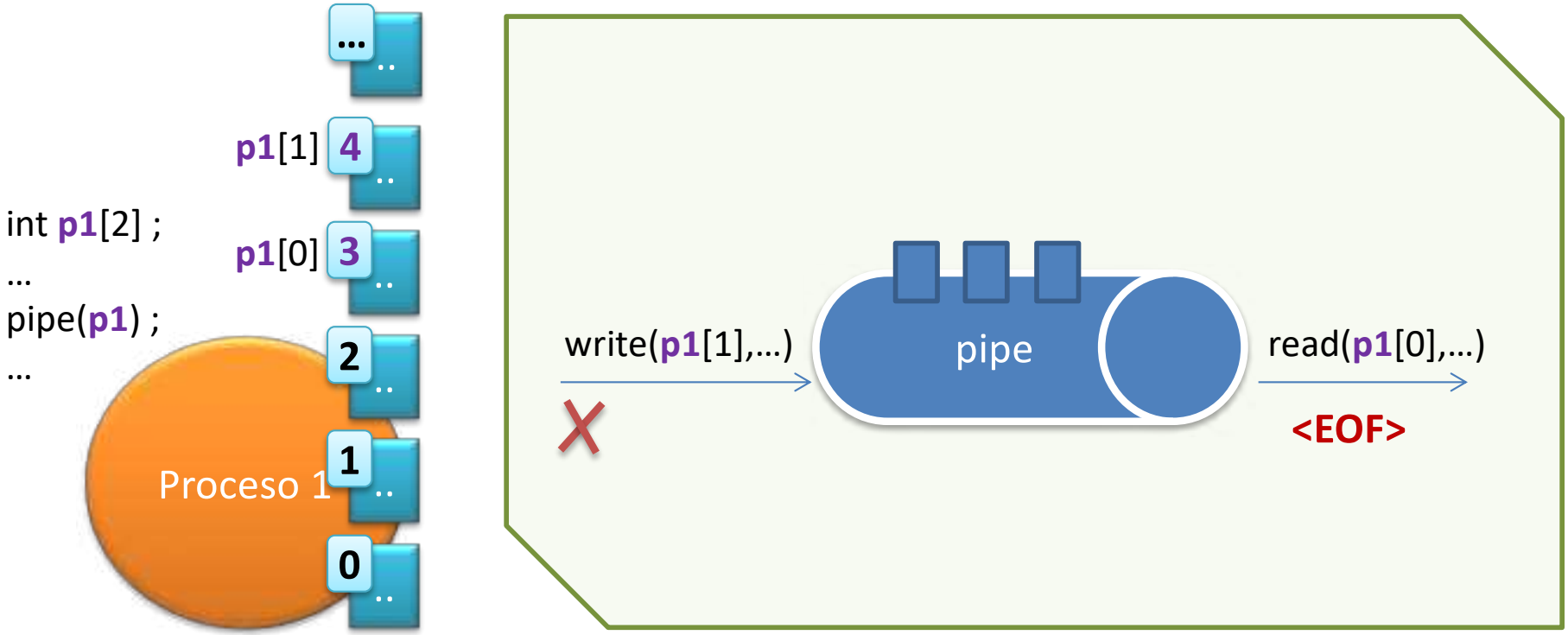


Tip

- Si se escribe en una tubería llena, se bloquea la ejecución del proceso hasta poder escribir.
- Si se lee de una tubería vacía, se bloquea la ejecución del proceso hasta poder leer algo.

Tubería

1 creación

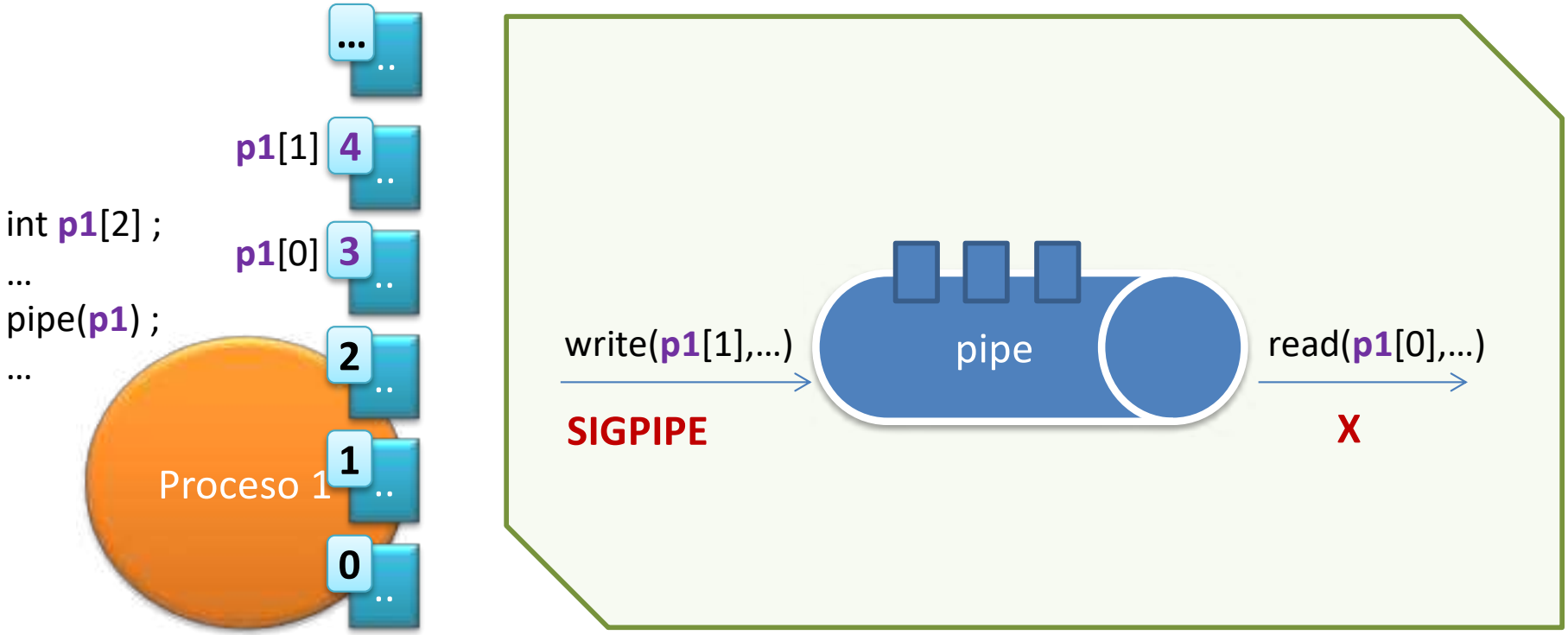


Tip

- Cuando todos los procesos escritores cierren la parte de escritura, entonces se manda un final de fichero (EOF) a los lectores.

Tubería

1 creación



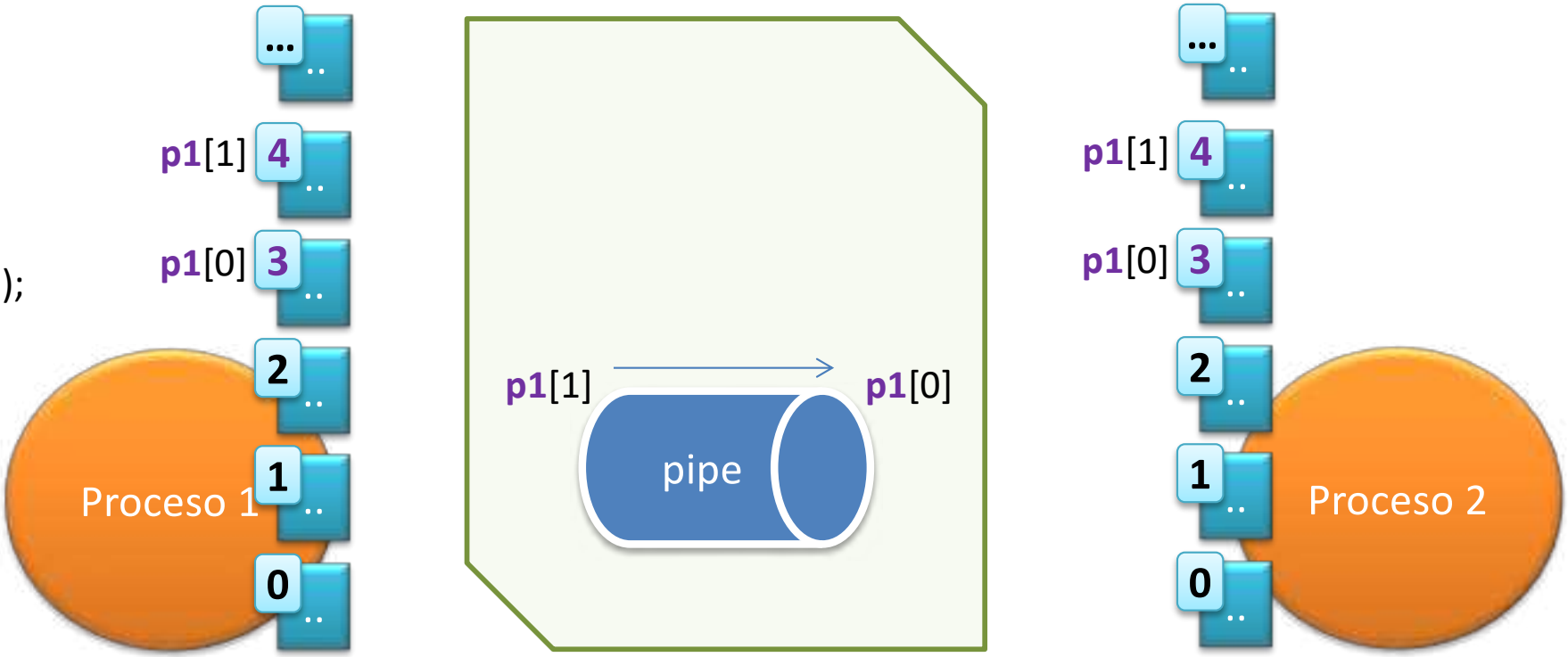
Tip

- Si se escribe en una tubería y no hay lectores el proceso escritor recibe la señal SIGPIPE.

Tubería

2 fork()

```
...
fork();
...
```

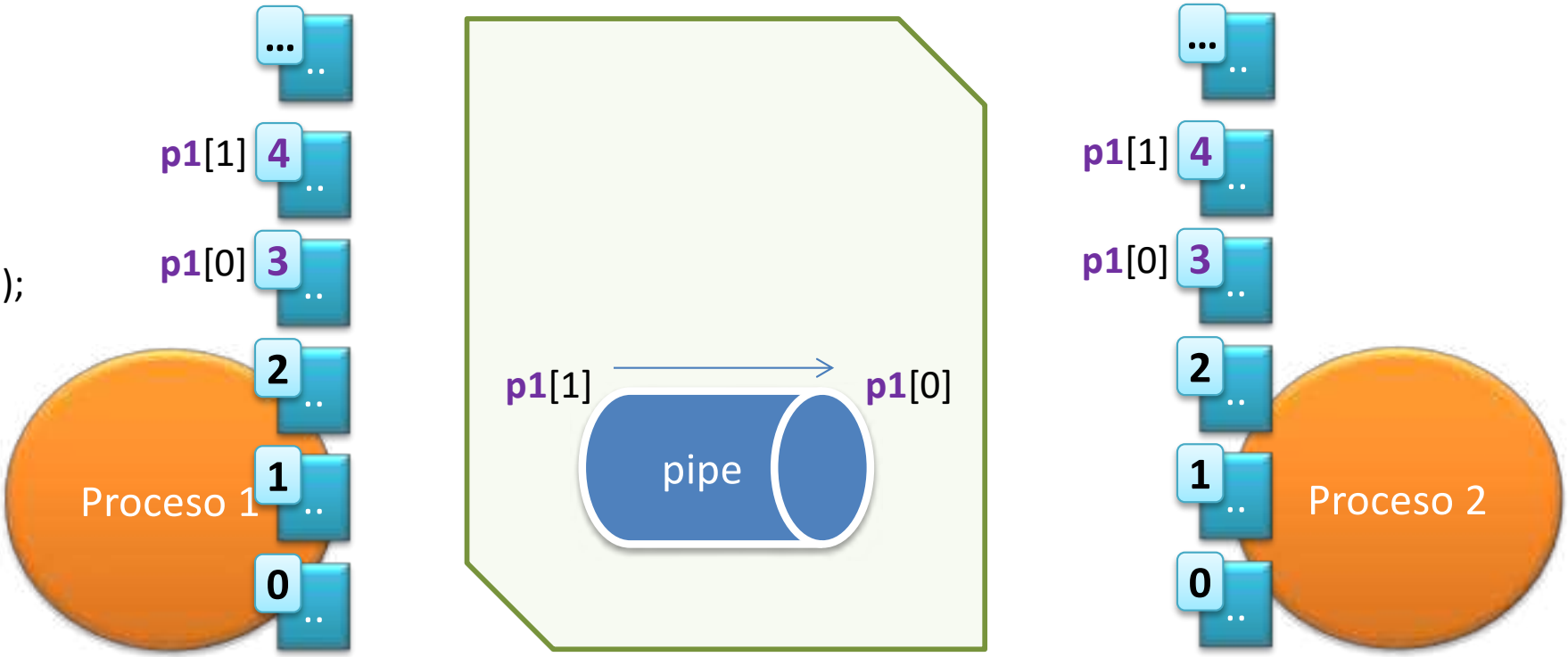


pipe() + fork() -> padre e hijo ven la misma tubería

Tubería

2 fork()

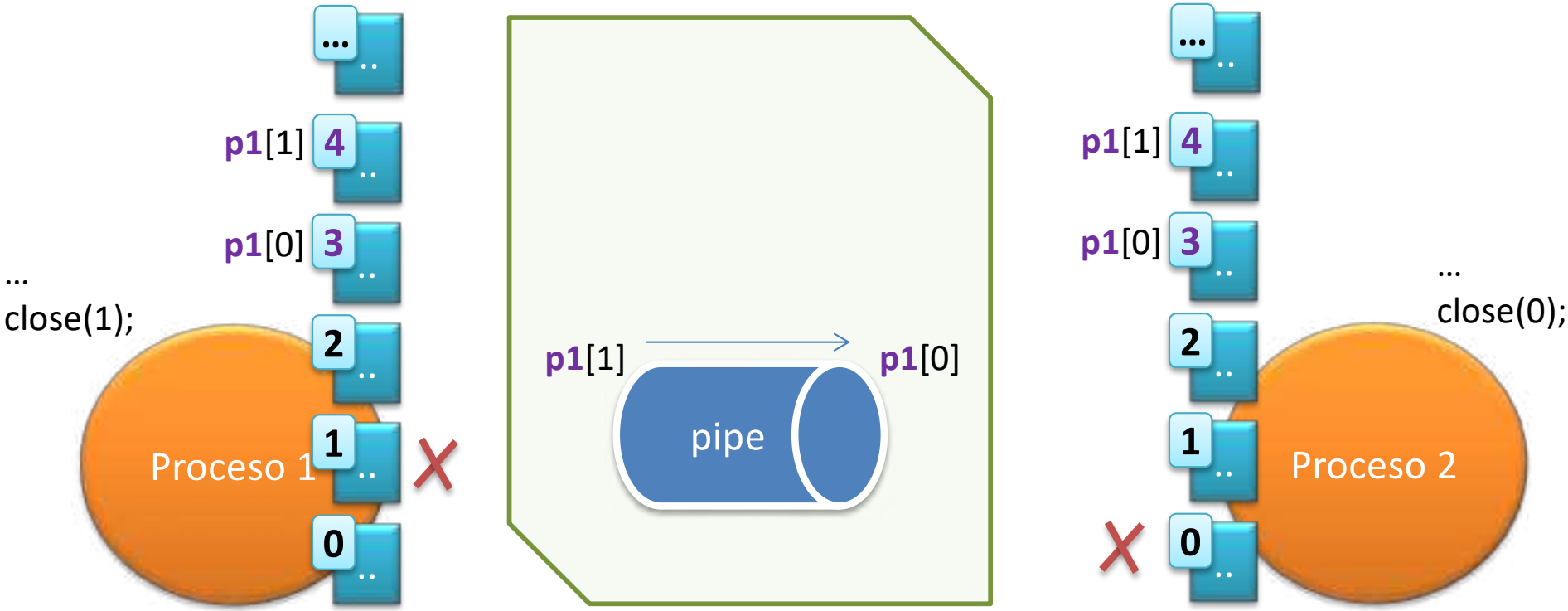
```
...
fork();
...
```



pipe() + fork() -> padre e hijo ven la misma tubería
-> ambos podrían leer y escribir en ella

Tubería

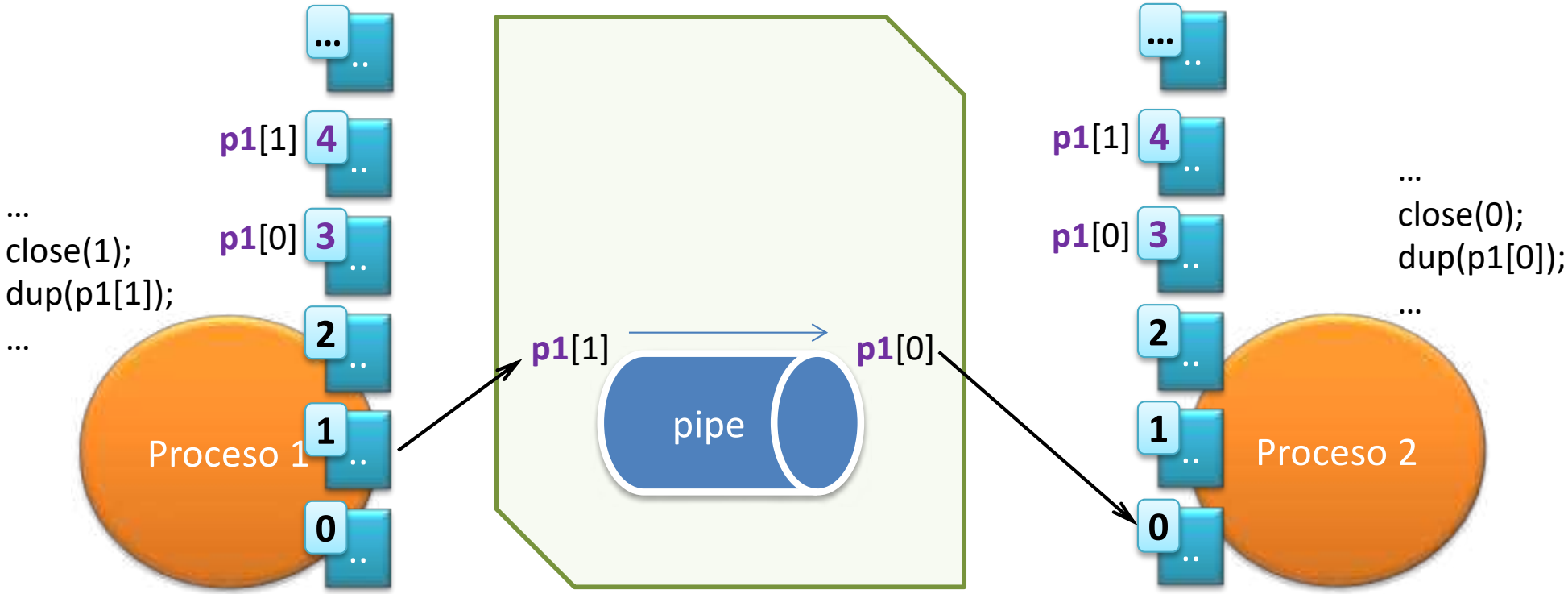
3 redirección



Redirección de la salida estándar en el padre…
Redirección de la entrada estándar en el hijo…

Tubería

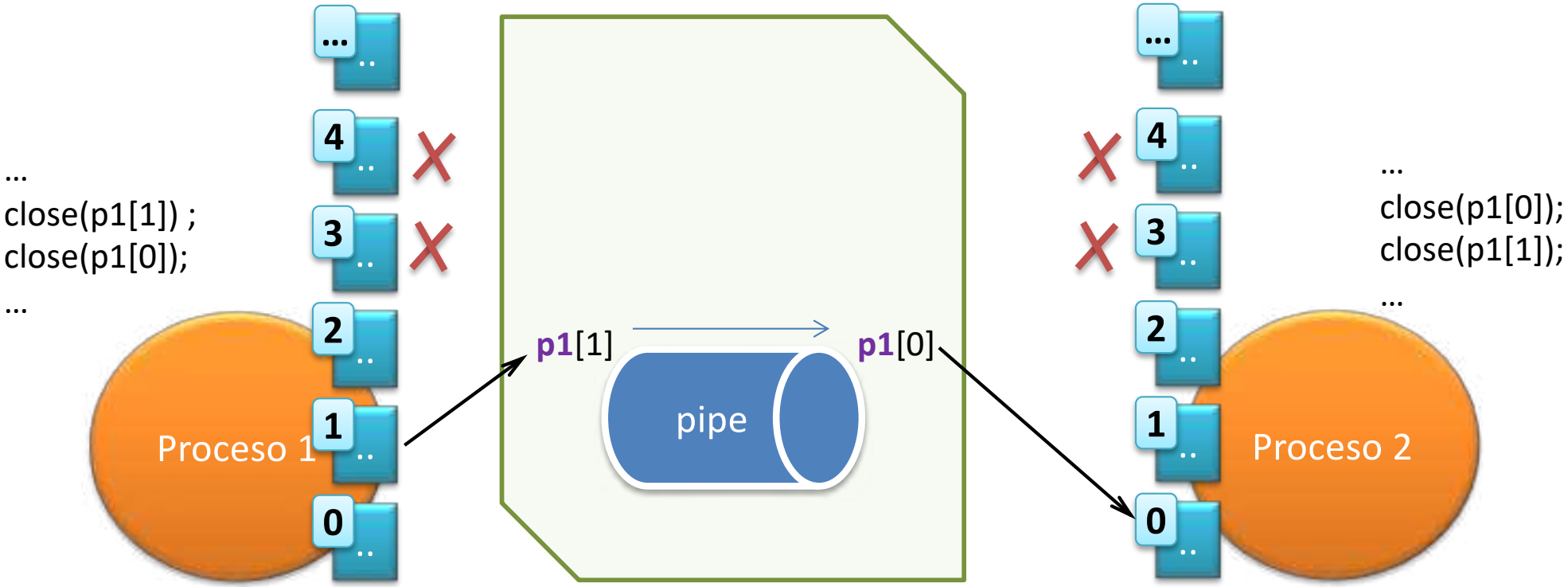
3 redirección



Redirección de la salida estándar en el padre…
Redirección de la entrada estándar en el hijo…

Tubería

4 limpieza

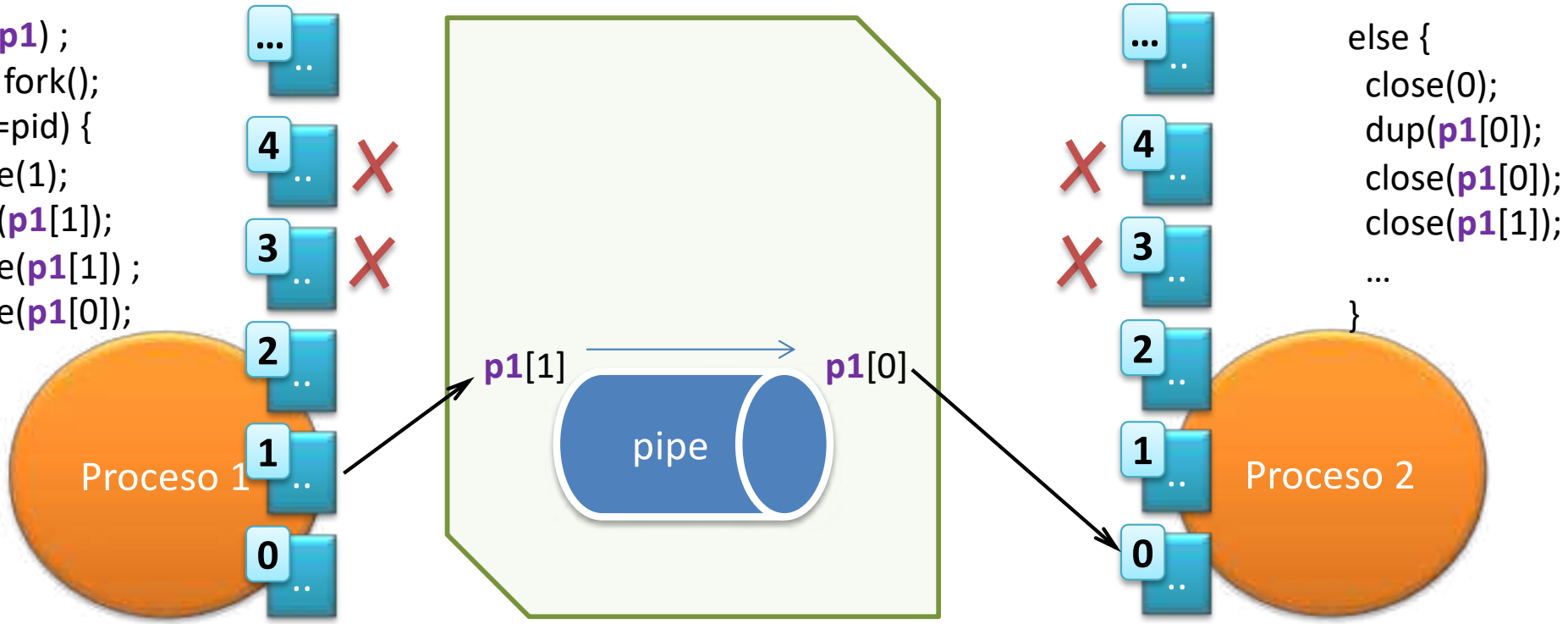


Cierre de los descriptores que no se usan en el padre…
Cierre de los descriptores que no se usan en el hijo…

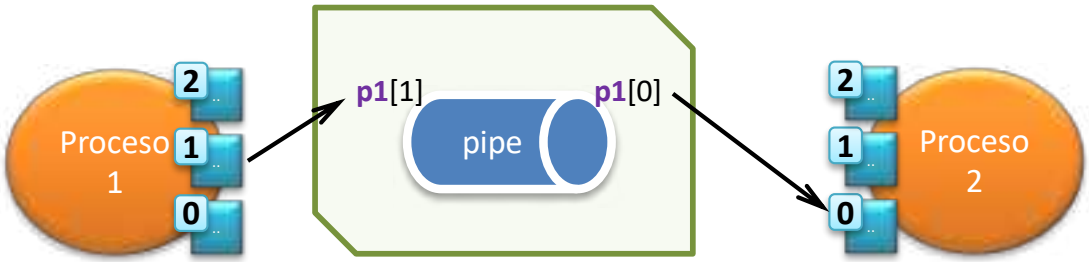
Tubería

resumen

```
int p1[2] ;
…
pipe(p1) ;
pid = fork();
if (0!=pid) {
  close(1);
  dup(p1[1]);
  close(p1[1]) ;
  close(p1[0]);
  …
}
```




```
else {
  close(0);
  dup(p1[0]);
  close(p1[0]);
  close(p1[1]);
  …
}
```




```
int p1[2] ;
…
pipe(p1) ;
pid = fork();
if (0!=pid) {
  close(1);
  dup(p1[1]);
  close(p1[1]) ;
  close(p1[0]);
  …
}
else {
  close(0);
  dup(p1[0]);
  close(p1[0]);
  close(p1[1]);
  …
}
```

1) Creación

2) fork()

3) Redirección (padre)

4) Limpieza (padre)

3) Redirección (hijo)

4) Limpieza (hijo)

Ejemplo: “ls | grep a”

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main (int argc, char *argv[])
{
  int fd[2];

  pipe(fd);
  if (fork()!=0) { /* código del padre */
     close(STDIN_FILENO);
     dup(fd[STDIN_FILENO]);
     close(fd[STDIN_FILENO]);
     close(fd[STDOUT_FILENO]);
     execlp(“grep", “grep", ”a”, NULL);
  } else { /* código del hijo */
     close(STDOUT_FILENO);
     dup(fd[STDOUT_FILENO]);
     close(fd[STDOUT_FILENO]);
     close(fd[STDIN_FILENO]);
     execlp("ls", "ls", NULL);
  }
  return 0;
}
```

Final del curso 2008-2009

Ejercicio 5 y 6 (3.5 puntos)

- Escribir una función en C sobre UNIX que cree tres procesos comunicados mediante una tubería, de manera que dos de ellos tengan la salida estándar asociada a la tubería y el otro la entrada estándar. *Argumentos: nombres de programa que deberán ejecutar los tres procesos hijos.*