

Lección 3

Señales, excepciones y pipes

Sistemas Operativos
Ingeniería Informática

Lecturas recomendadas

Base



1. Carretero 2020:
 1. Cap. 5
2. Carretero 2007:
 1. Cap. 3.6 y 3.7
Cap. 3.9 y 3.13

Recomendada



1. Tanenbaum 2006:
 1. (es) Cap. 2.2
 2. (en) Cap.2.1.7
2. Stallings 2005:
 1. 4.1, 4.4, 4.5 y 4.6
3. Silberschatz 2006:
 1. 4

¡ATENCIÓN!

- ❑ Este material es un guión de la clase pero no son los apuntes de la asignatura.
- ❑ Los libros dados en la bibliografía junto con lo explicado en clase representa el material de estudio para el temario de la asignatura.

Contenidos

1. Señales y excepciones.
2. Temporizadores.
3. Entorno de un proceso.
4. **Comunicación de procesos con tuberías (pipes).**
 - ▶ Paso de mensajes local.

Contenidos

1. Señales y excepciones.
2. Temporizadores.
3. Entorno de un proceso.
4. **Comunicación de procesos con tuberías (pipes).**
 - ▶ Paso de mensajes local.

Ejemplo redirección **entrada**

- **Linux: redirección y tuberías**
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

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

f1.txt

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

grep ocho < f1

Ejemplo redirección salida

- **Linux: redirección y tuberías**
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

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

f1.txt

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

siete, ocho, nueve

grep ocho < f1 > s1

Ejemplo redirección salida

- **Linux: redirección y tuberías**
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

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

f1.txt

Dependiente del
intérprete de
mandatos usado

siete, ocho, nueve

grep ocho f1 1> s1

Ejemplo redirección **error**

- **Linux: redirección y tuberías**
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - *fork()*
- Tuberías

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

f1.txt

Dependiente del
intérprete de
mandatos usado

grep: f2: No existe el archivo o el directorio

grep ocho xx 2> s1

Ejemplo de uso de tuberías

- **Linux: redirección y tuberías**
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

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

f1.txt

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

cat f1 | head -3 | tail -1

siete, ocho, nueve

Descriptores de ficheros

- Linux: redirección y tuberías
- **Los descriptores de ficheros**
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías



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

- Linux: redirección y tuberías
- **Los descriptores de ficheros**
 - Redirección
 - Duplicación
 - *fork()*
- Tuberías

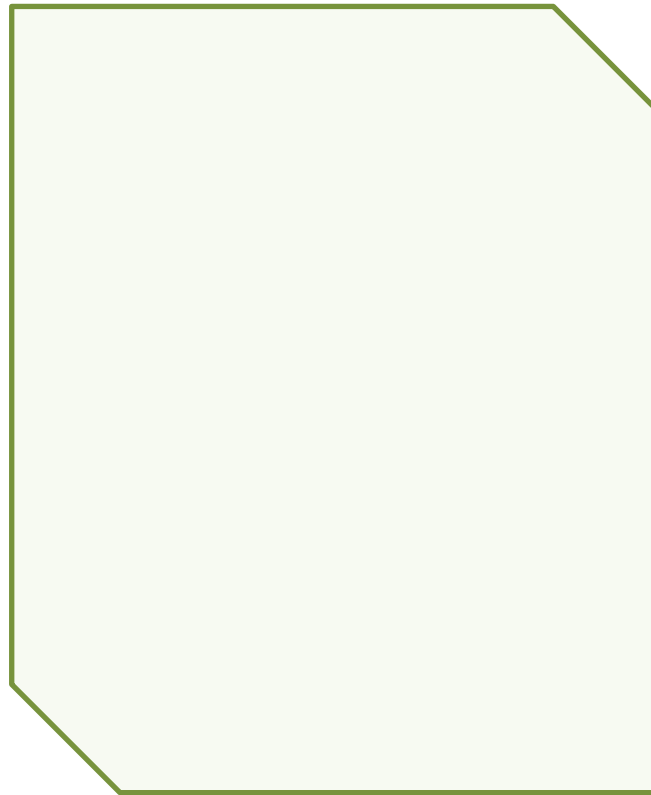


Por defecto se utilizan los tres primeros para la entrada estándar, salida estándar y salida de error respectivamente.

Descriptores de ficheros

abstracción ofrecida

- Linux: redirección y tuberías
- **Los descriptores de ficheros**
 - Redirección
 - Duplicación
 - *fork()*
- Tuberías

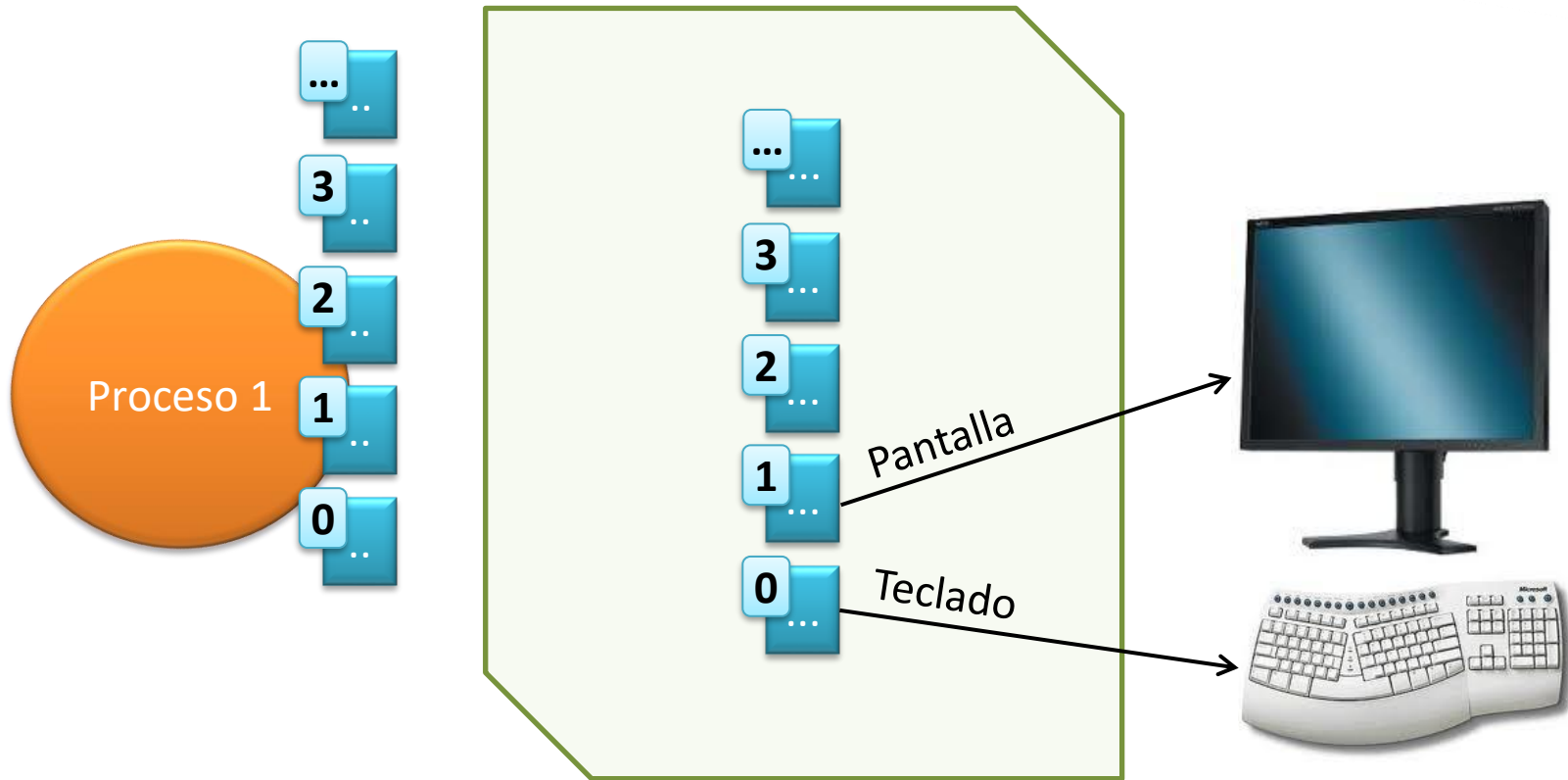


Los descriptores de ficheros son una abstracción ofrecida por el sistema operativo para referenciar los dispositivos reales. Igual que una llave numerada para una consigna.

Descriptores de ficheros

abstracción ofrecida

- Linux: redirección y tuberías
- **Los descriptores de ficheros**
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

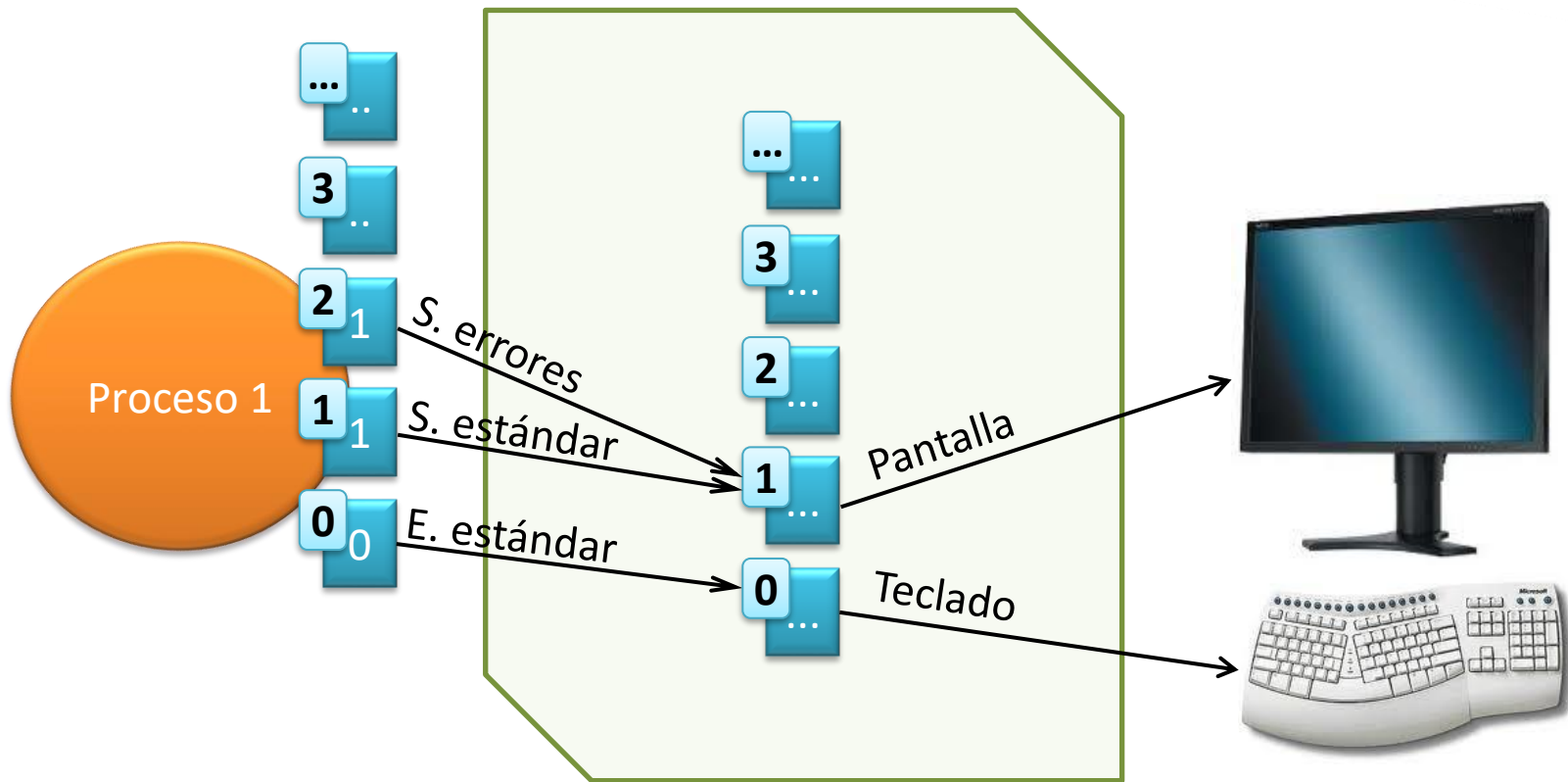


El sistema operativo mantiene una tabla interna con la información real de contacto con los dispositivos y ficheros con los que los procesos piden comunicarse...

Descriptores de ficheros

abstracción ofrecida

- Linux: redirección y tuberías
- **Los descriptores de ficheros**
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

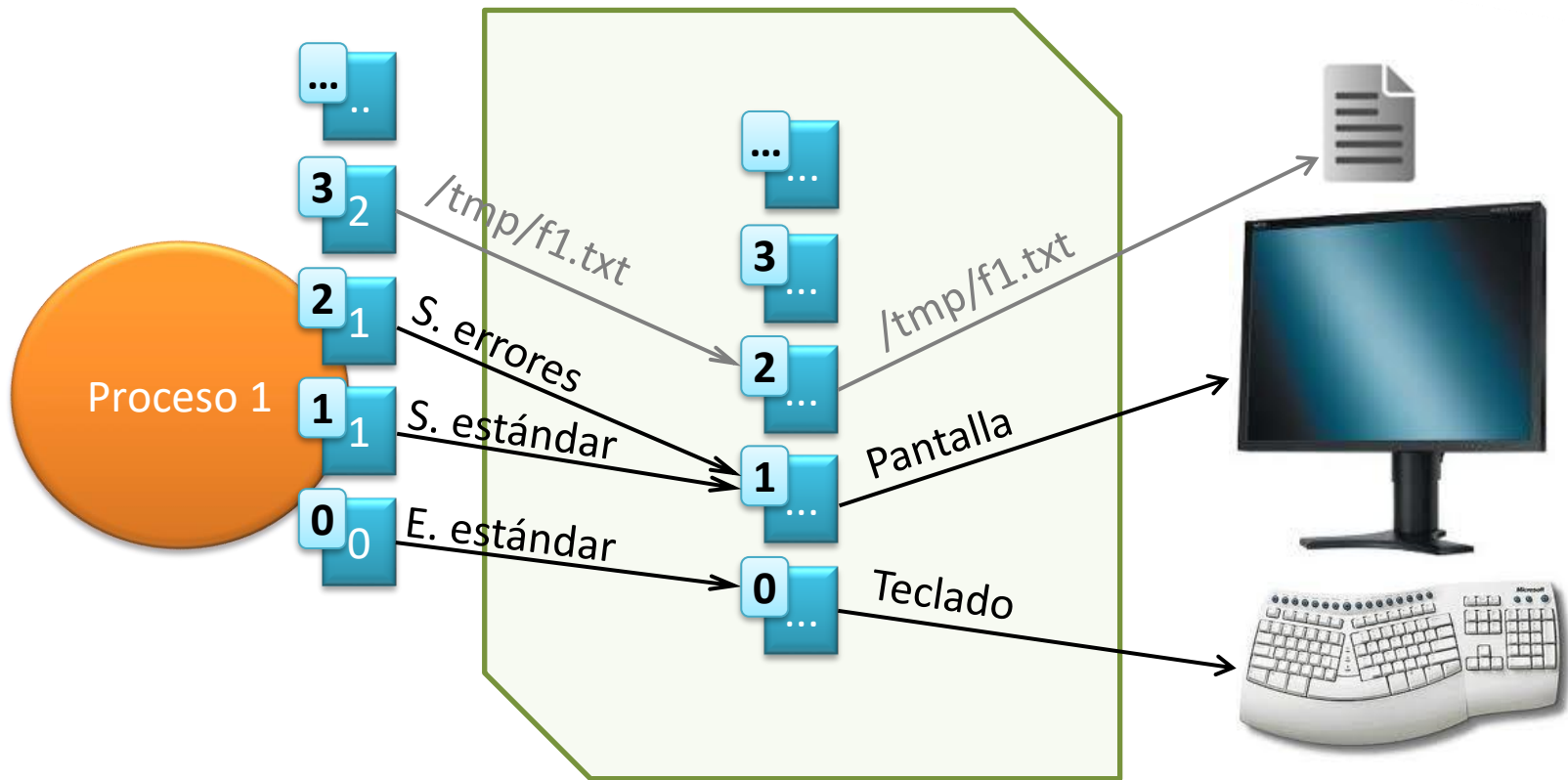


...Y los descriptores de ficheros son el índice de la tabla que hay por proceso, cuyo contenido es a su vez el índice de la tabla interna del sistema operativo.

Descriptores de ficheros

abstracción ofrecida

- Linux: redirección y tuberías
- **Los descriptores de ficheros**
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

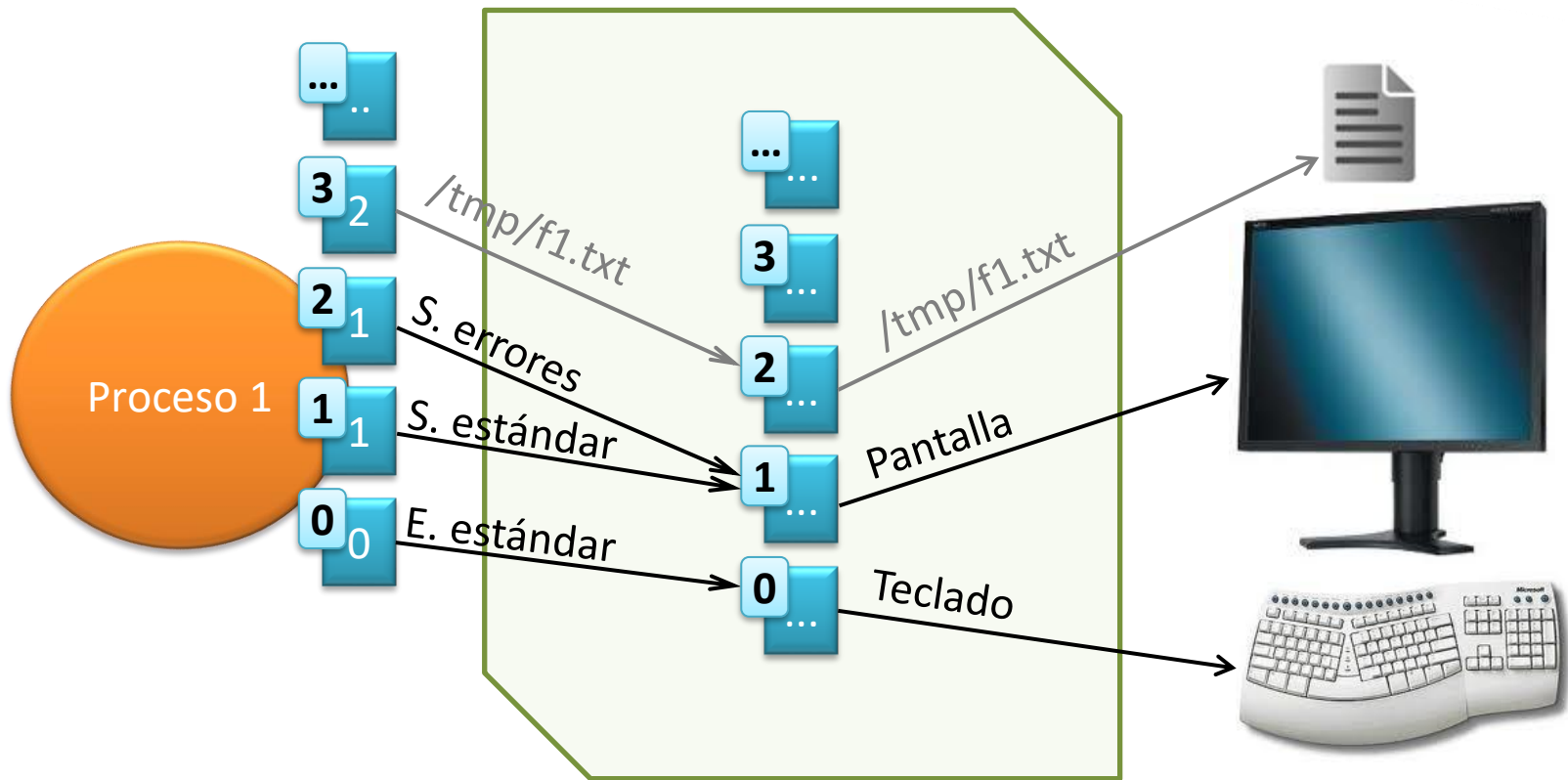


Cuando se pide un nuevo descriptor de ficheros (al abrir un fichero) se busca el primero hueco libre de la tabla y el índice de esa posición es el descriptor asignado.

Descriptores de ficheros

redirección a fichero

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías



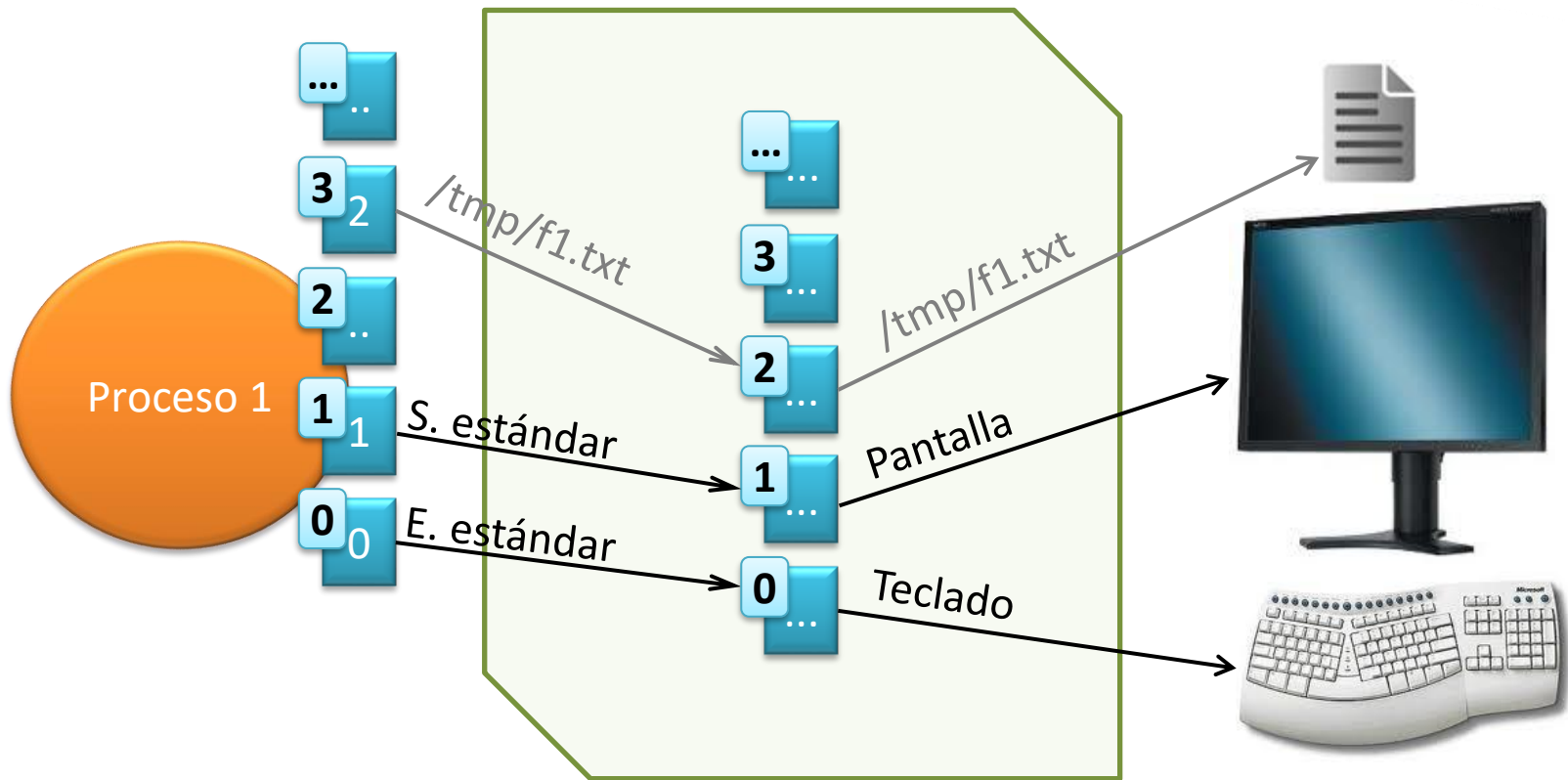
1. `close(2);`
2. `open("/tmp/errores.txt");`



Descriptores de ficheros

redirección a fichero

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

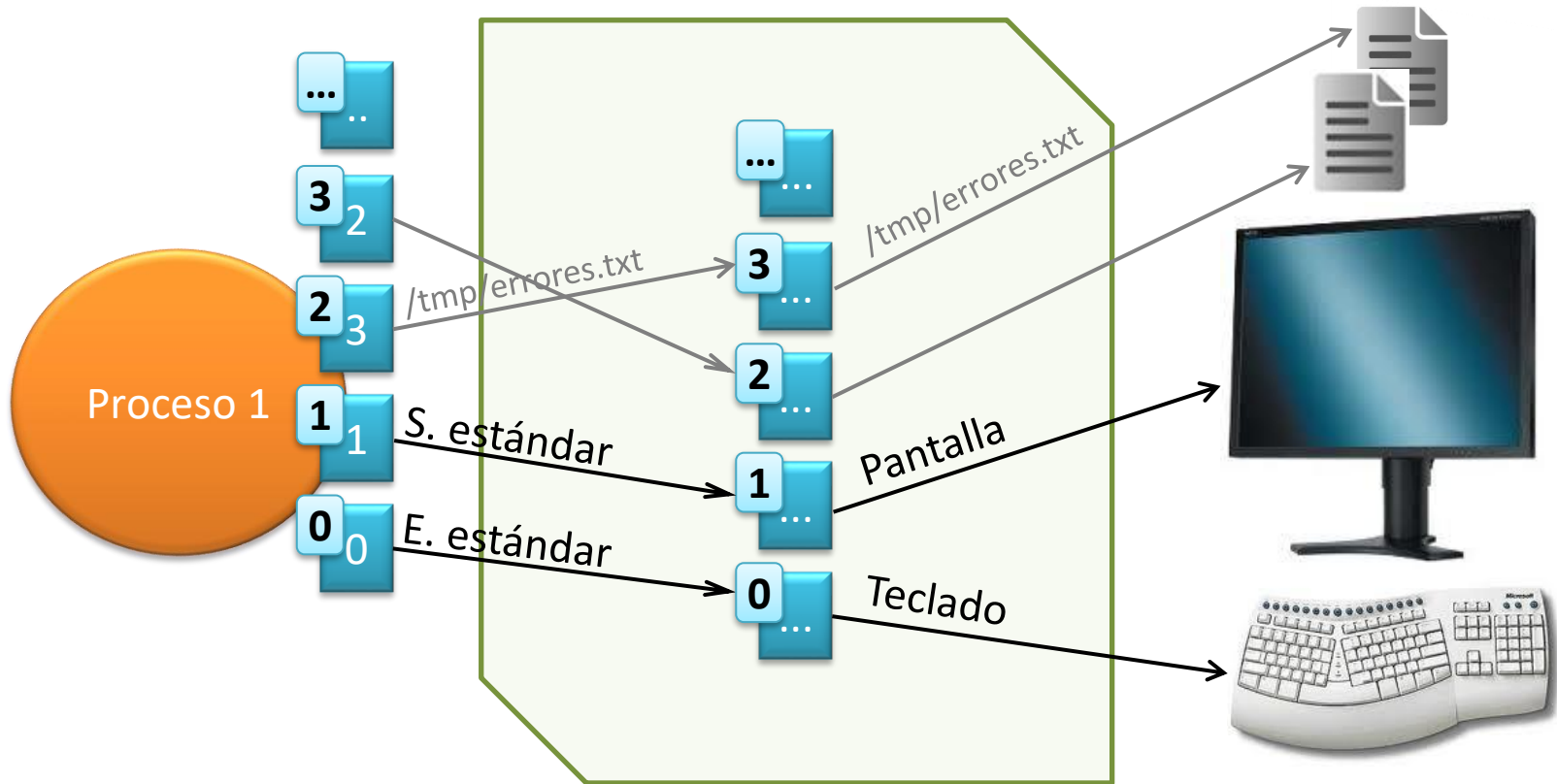


1. `close(2);`
2. `open("/tmp/errores.txt");`

Descriptores de ficheros

redirección a fichero

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

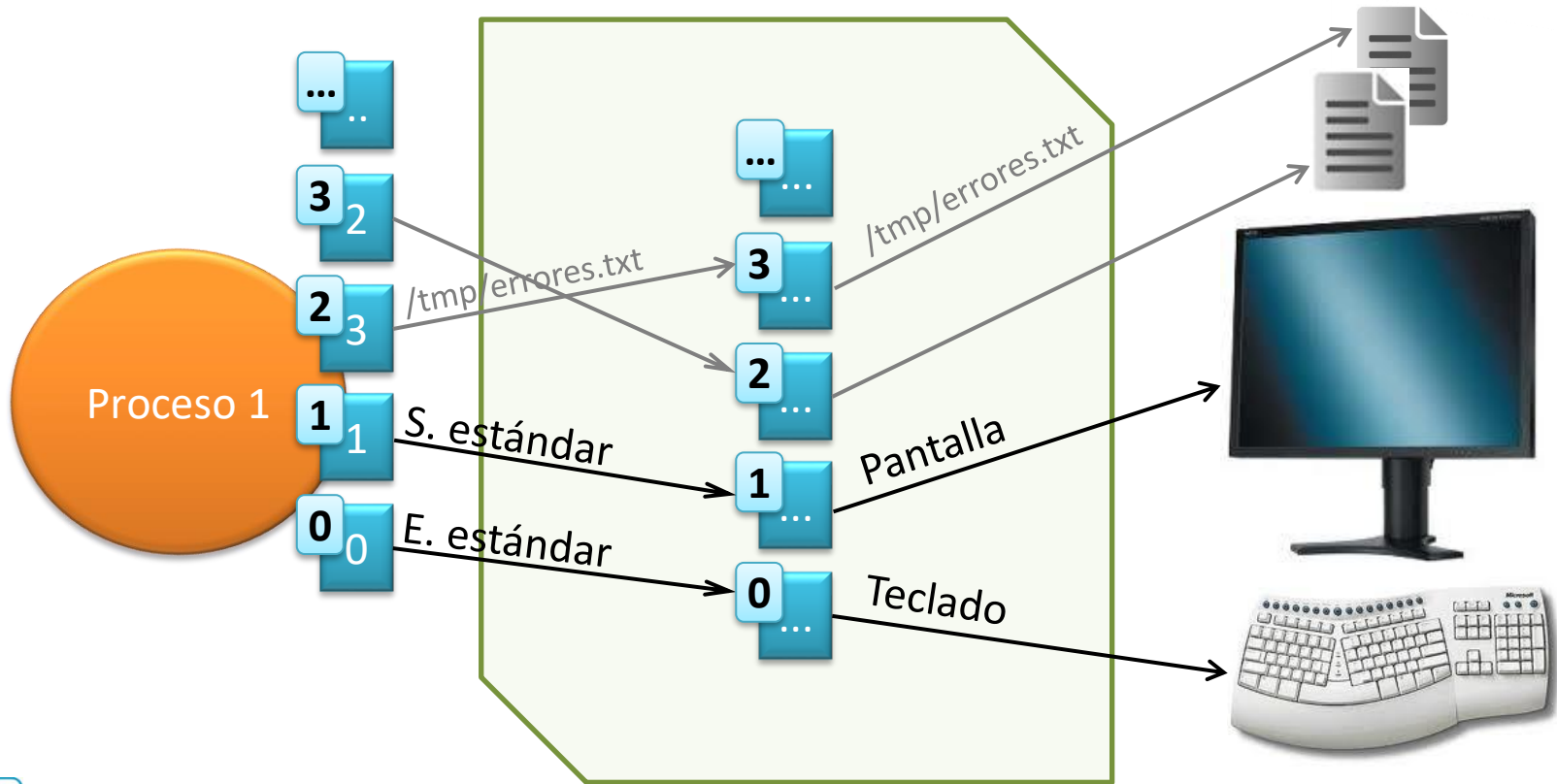


1. `close(2);`
2. `open("/tmp/errores.txt");`

Descriptores de ficheros

redirección a fichero

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

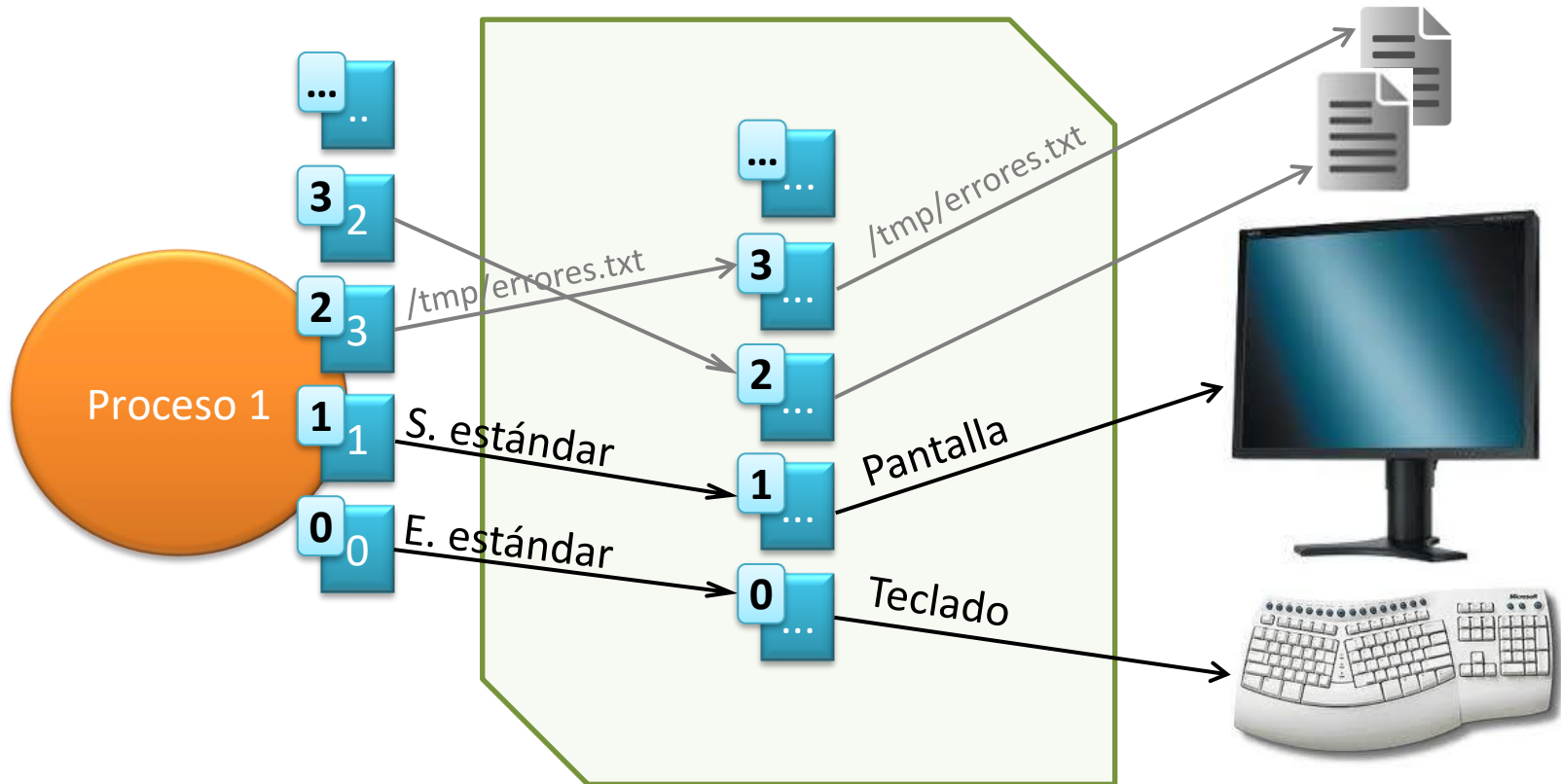


`close(2) + open("/tmp/errores.txt")`
Es posible cambiar el archivo asociado a un descriptor.

Descriptores de ficheros

duplicación de descriptor

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - **Duplicación**
 - `fork()`
- Tuberías

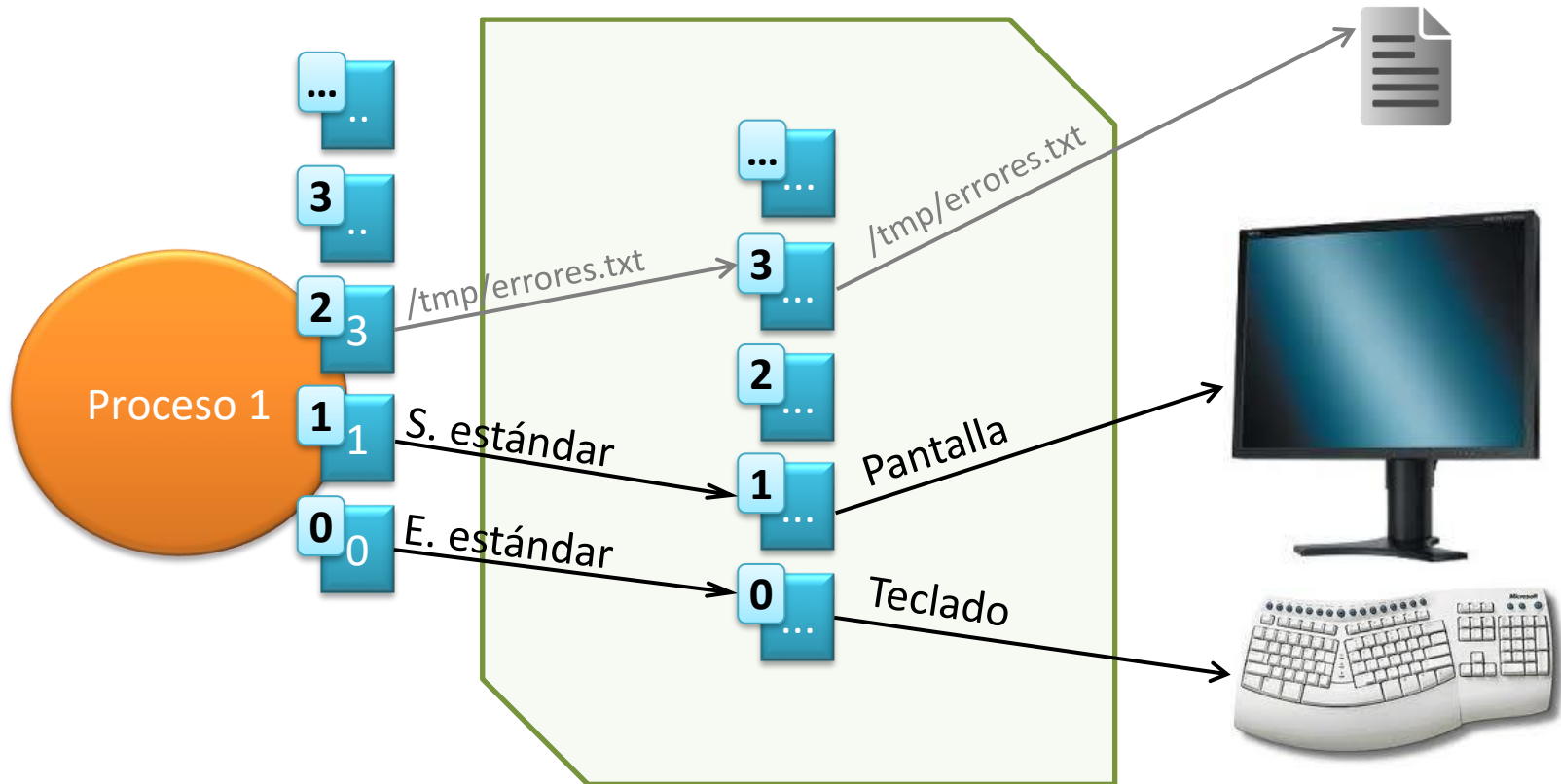


¿ 1. `close(3);` ?
2. `dup(2);` ?

Descriptores de ficheros

duplicación de descriptor

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - **Duplicación**
 - `fork()`
- Tuberías

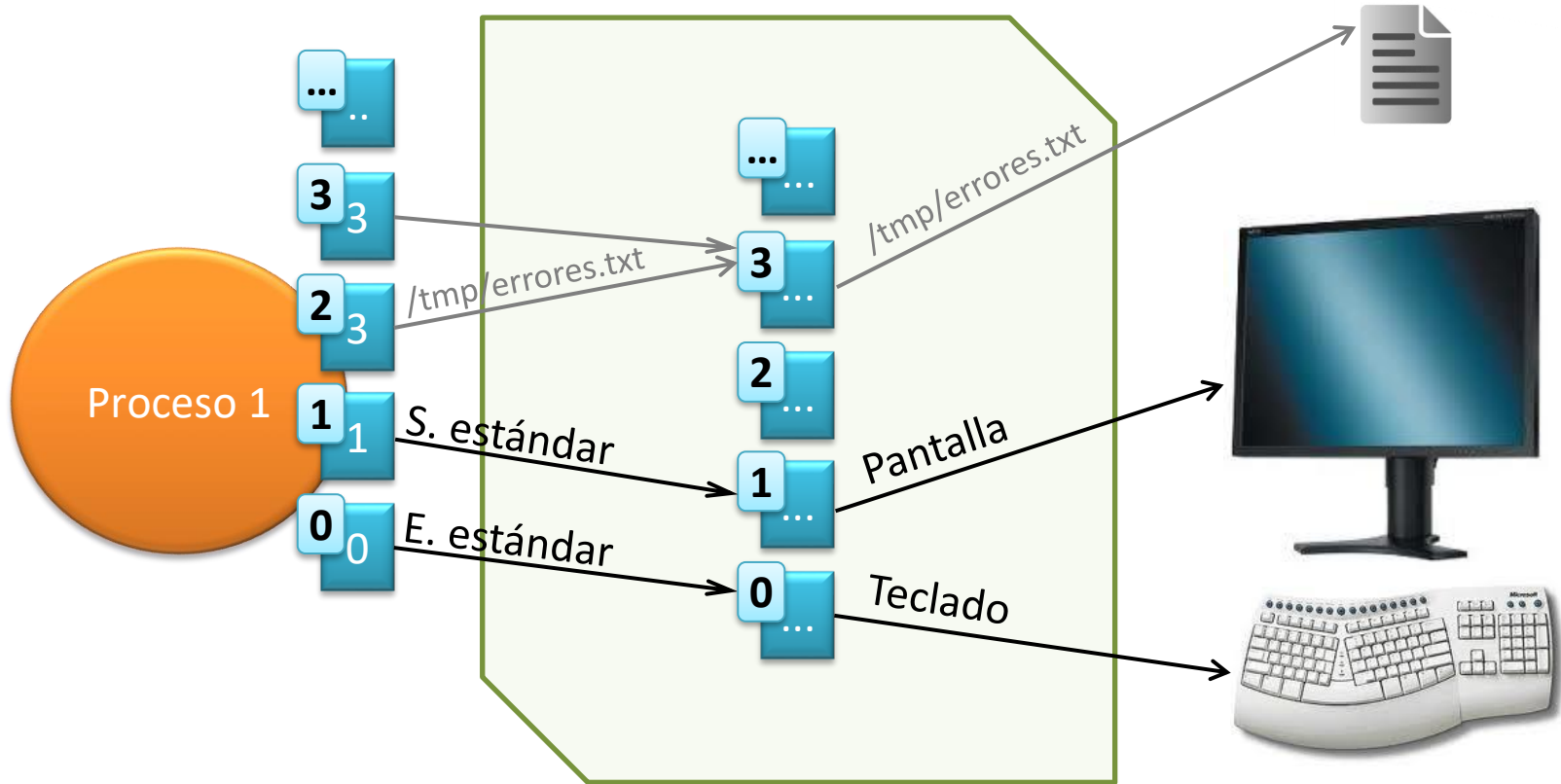


1. `close(3);`
2. `dup(2);`

Descriptores de ficheros

duplicación de descriptor

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - **Duplicación**
 - `fork()`
- Tuberías

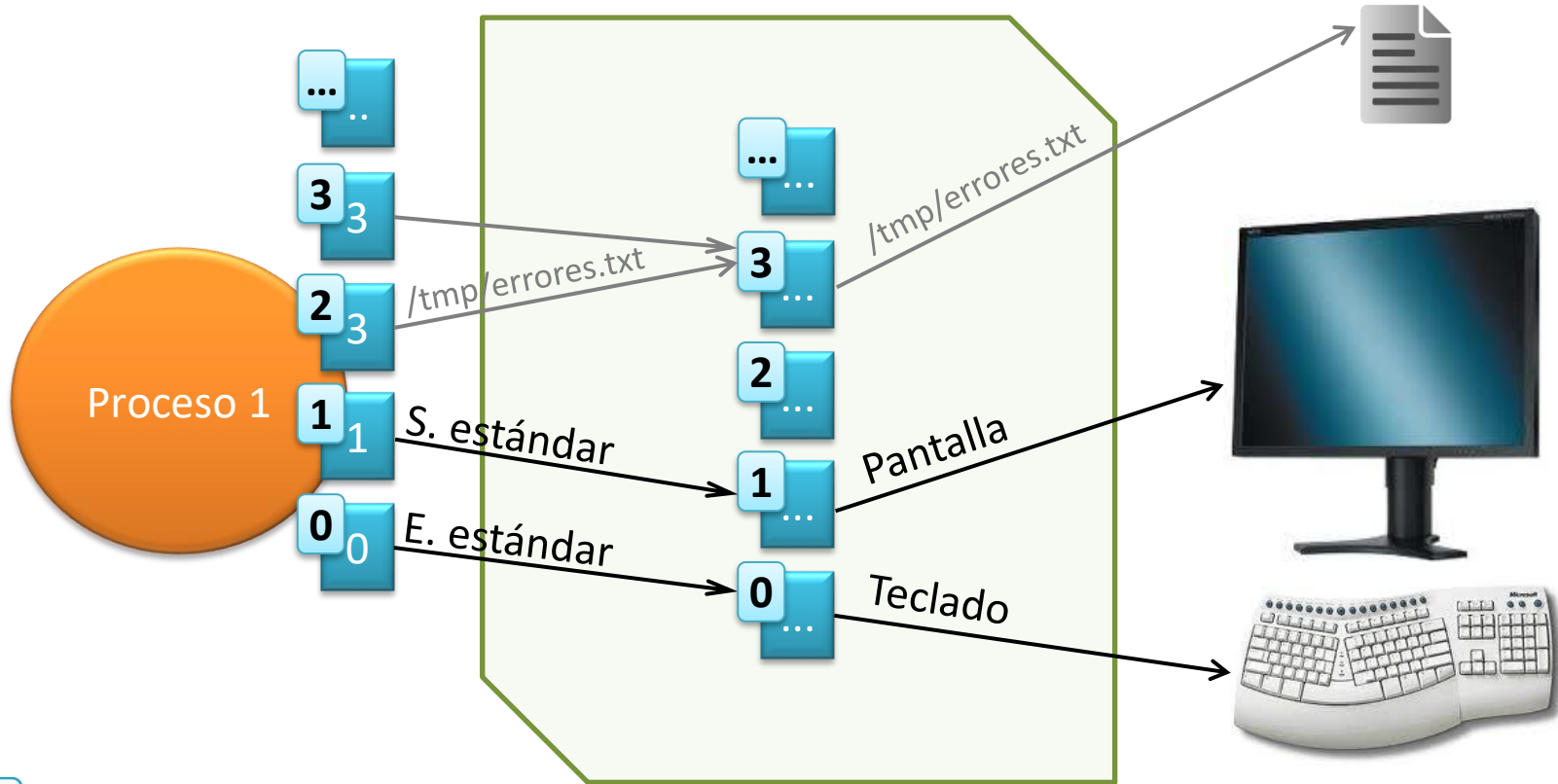


1. `close(3);`
2. **`dup(2);`**

Descriptores de ficheros

duplicación de descriptor

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - **Duplicación**
 - `fork()`
- Tuberías



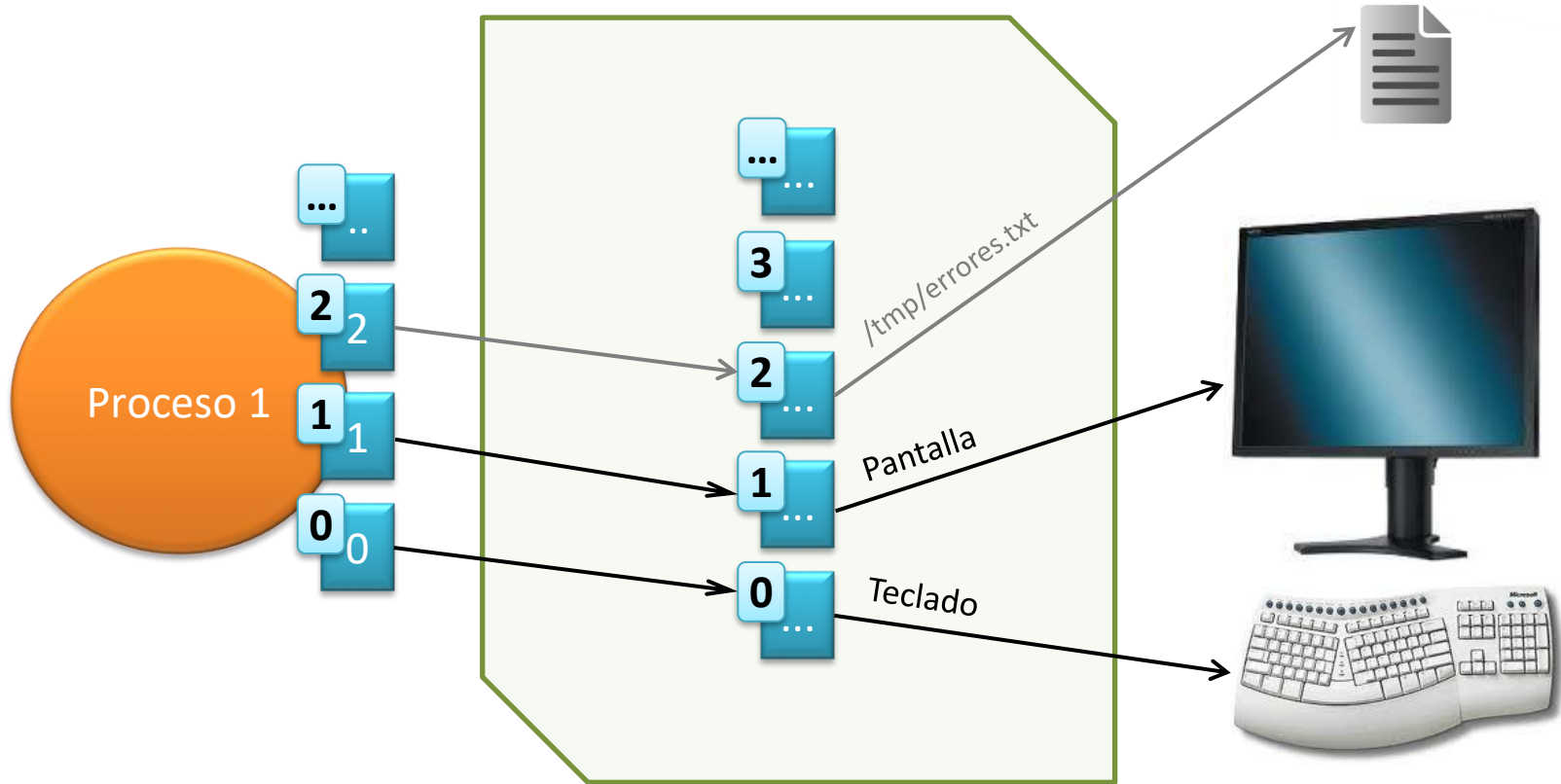
`close(3) + dup(2)`

Permite acceder a un mismo fichero desde dos descriptores diferentes

Descriptores de ficheros

llamada `fork()`

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - **`fork()`**
- Tuberías

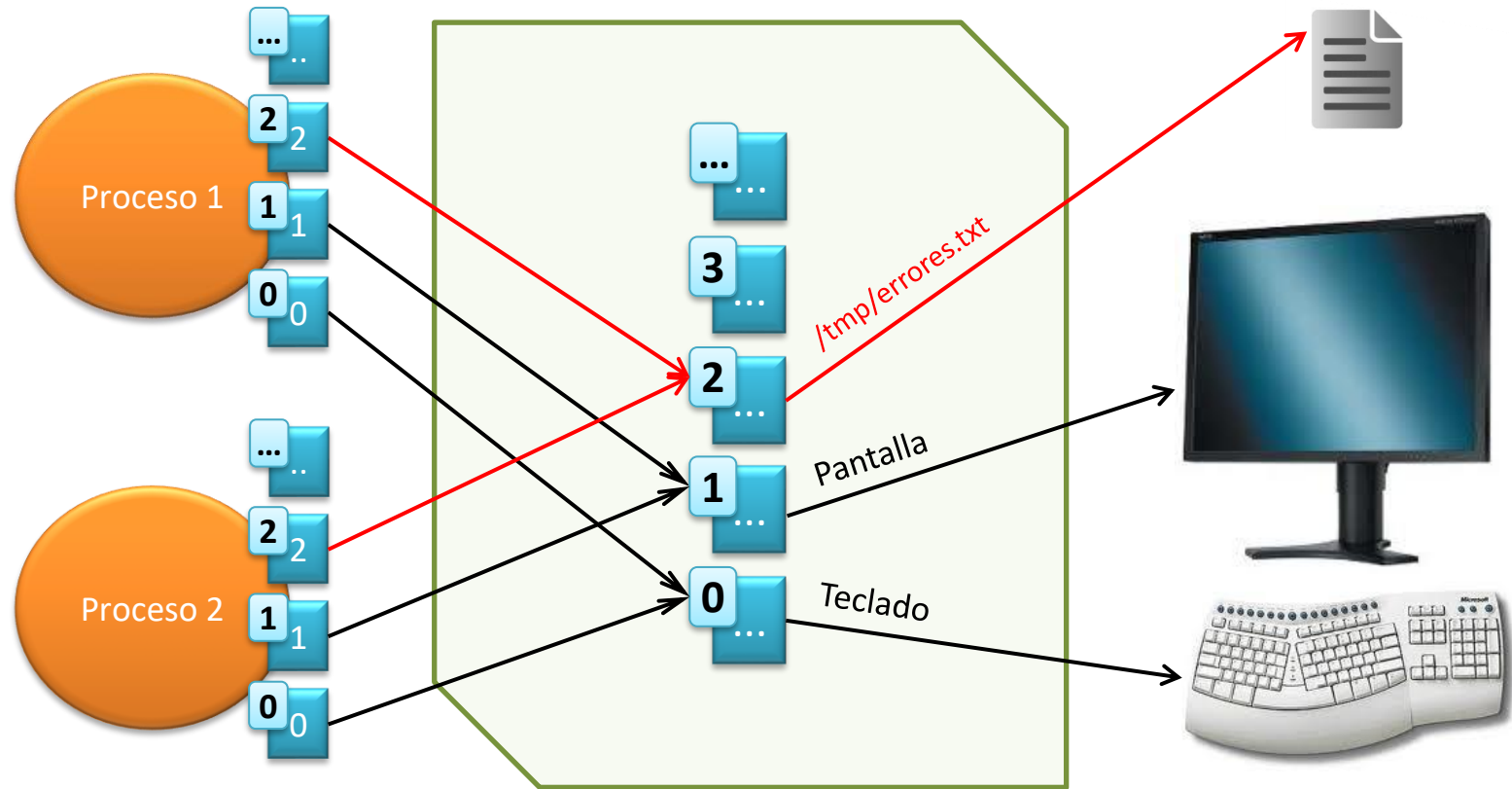


`fork()` crea un duplicado del hijo

Descriptores de ficheros

llamada `fork()`

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

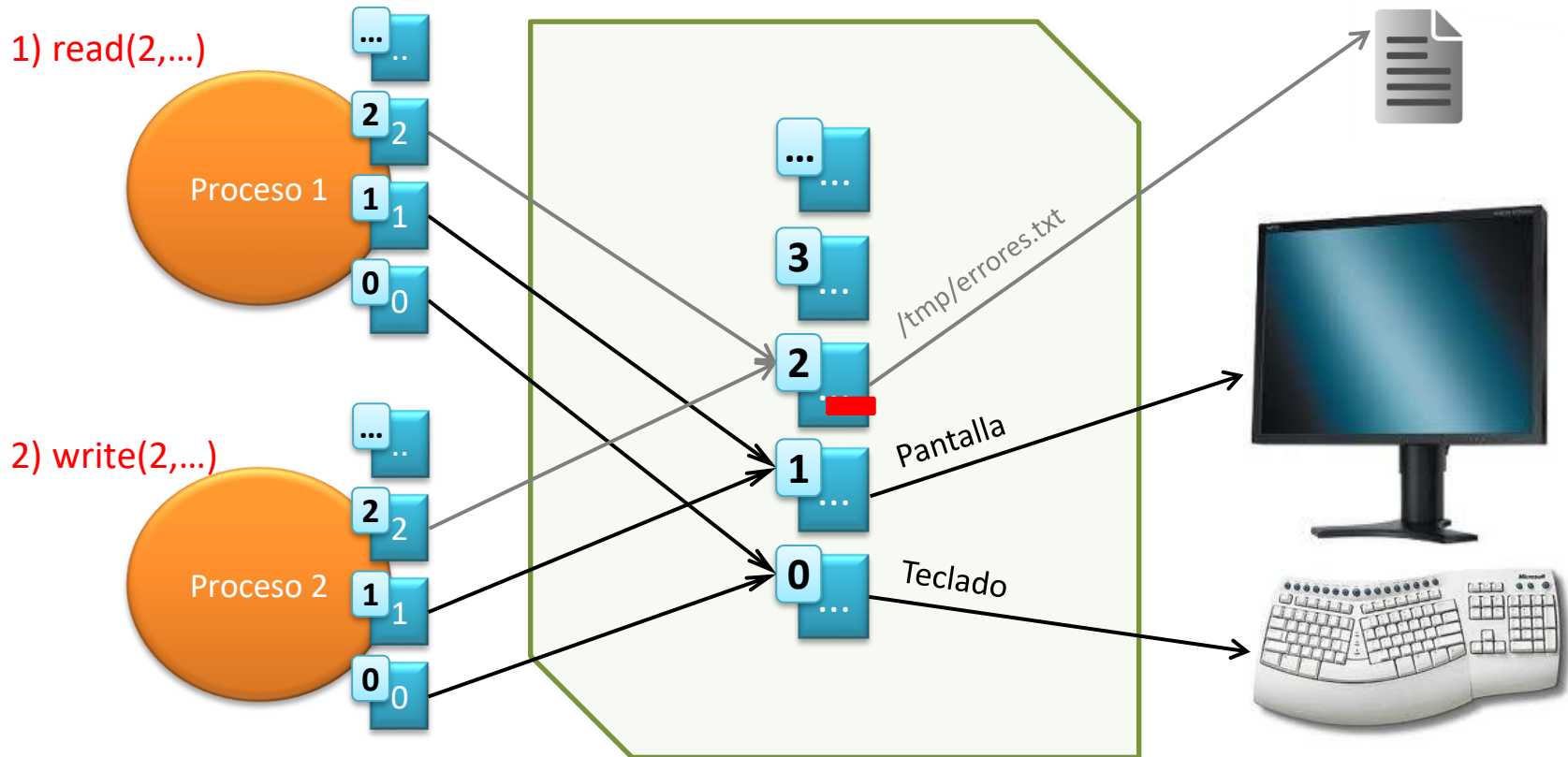


- **Ambos tienen descriptores iguales (redirecciones antes del `fork()` se heredan)**
- Ambos referencian los mismos elementos (posición L/E después del `fork()` común)

Descriptores de ficheros

llamada `fork()`

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías



- 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

1.- creación

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2];
```

```
...
```

```
pipe(p1);
```

```
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

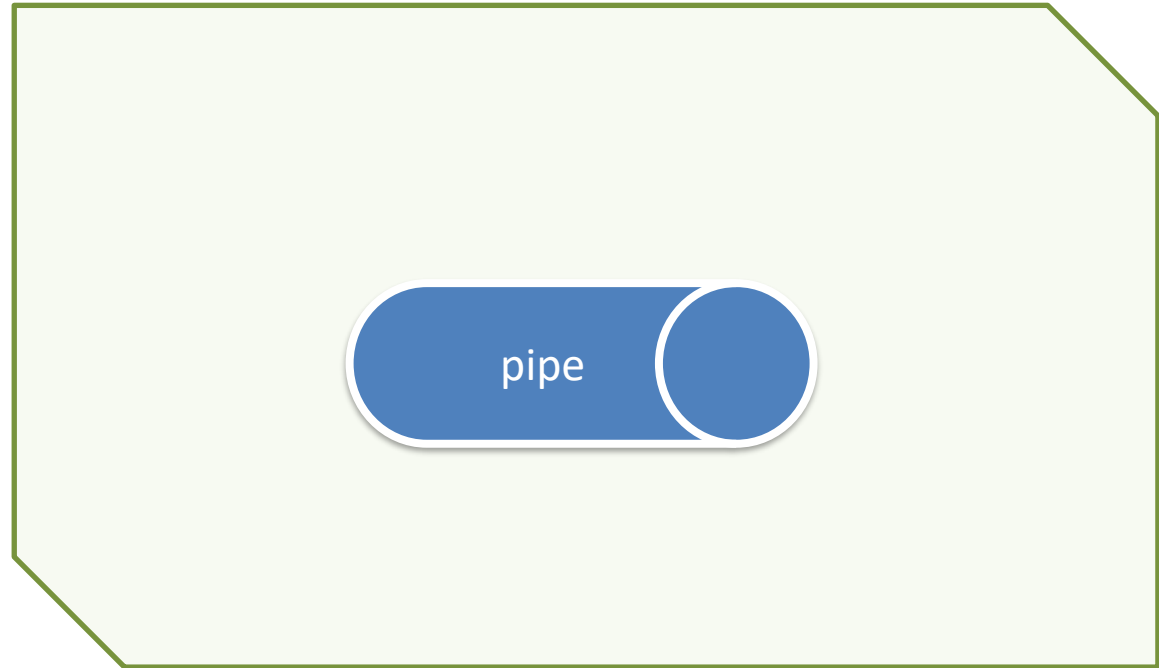
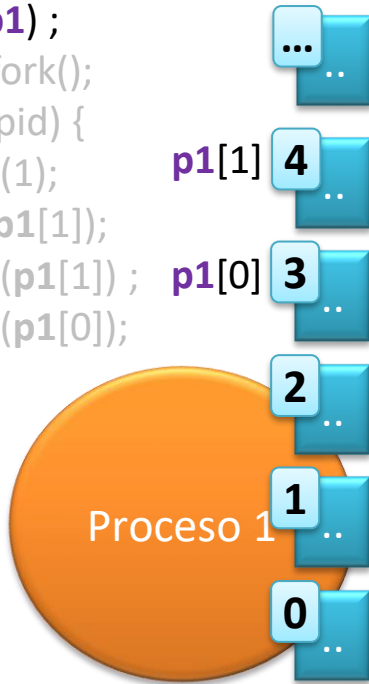
```
    dup(p1[1]);
```

```
    close(p1[1]);
```

```
    close(p1[0]);
```

```
    ...
```

```
}
```



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

1.- creación

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2] ;
```

...

```
pipe(p1) ;
```

```
pid = fork();
```

```
if (0!=pid) {
```

```
close(1);
```

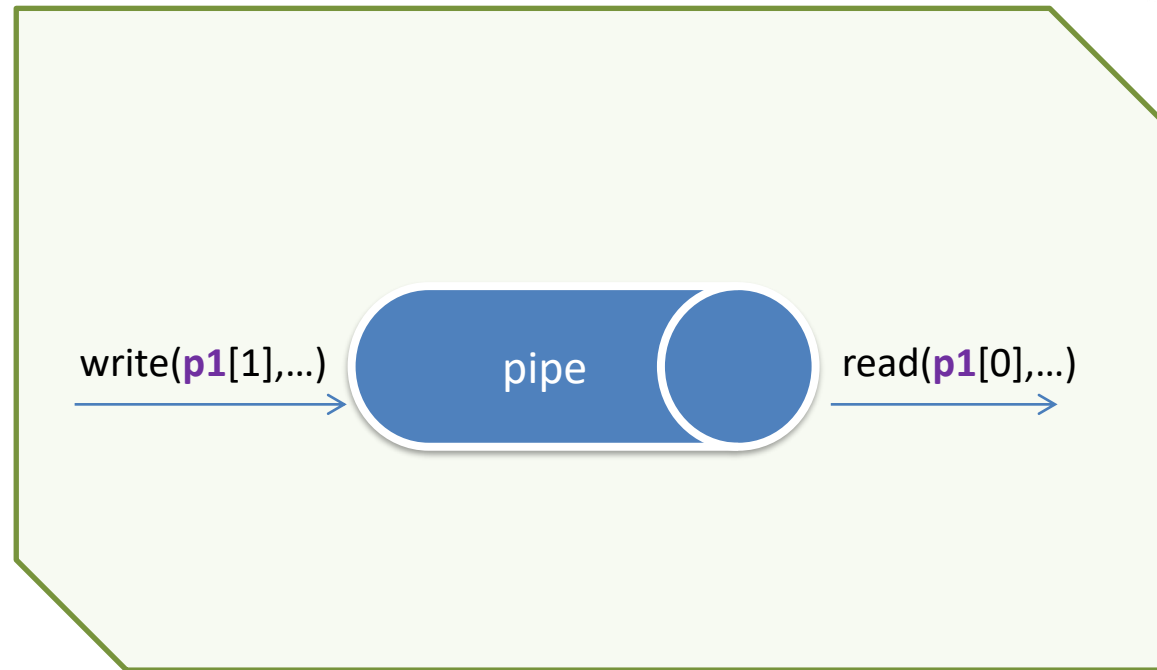
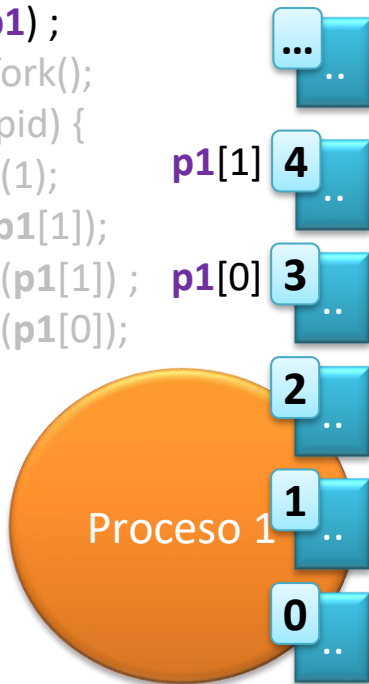
```
dup(p1[1]);
```

```
close(p1[1]) ; p1[0]
```

```
close(p1[0]);
```

...

}



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

Descriptores de ficheros

1.- creación

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2];
```

```
...
```

```
pipe(p1);
```

```
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

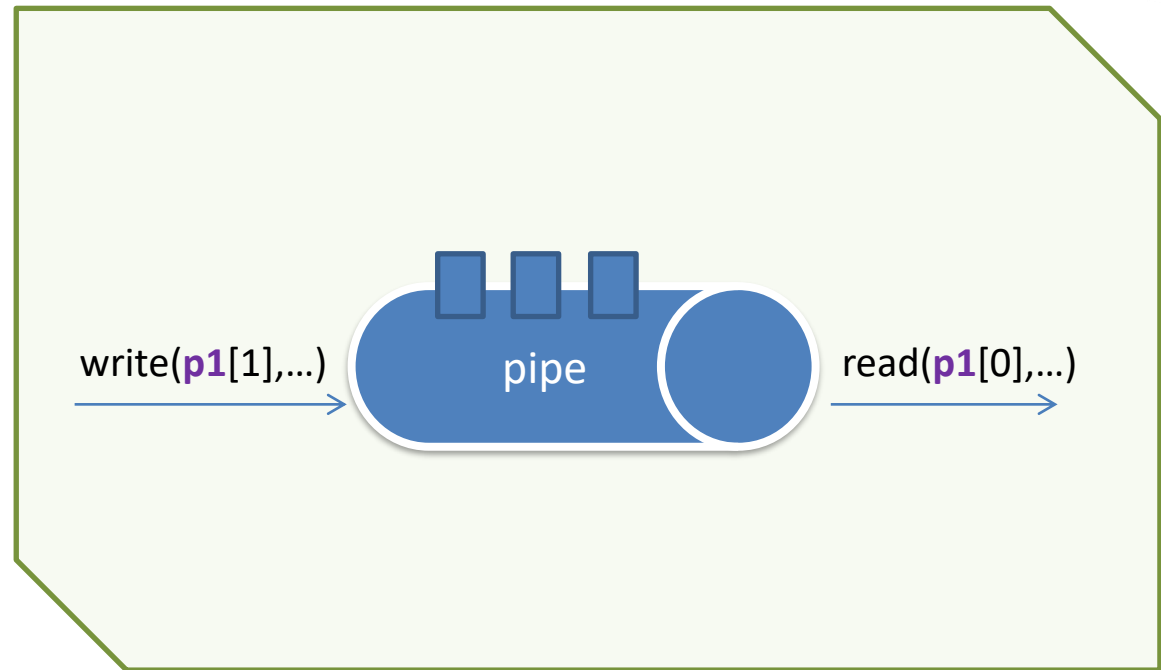
```
    dup(p1[1]);
```

```
    close(p1[1]);
```

```
    close(p1[0]);
```

```
    close(p1[0]);
```

```
...}
```



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.

Descriptores de ficheros

1.- creación

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2];
```

```
...
```

```
pipe(p1);
```

```
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

```
    dup(p1[1]);
```

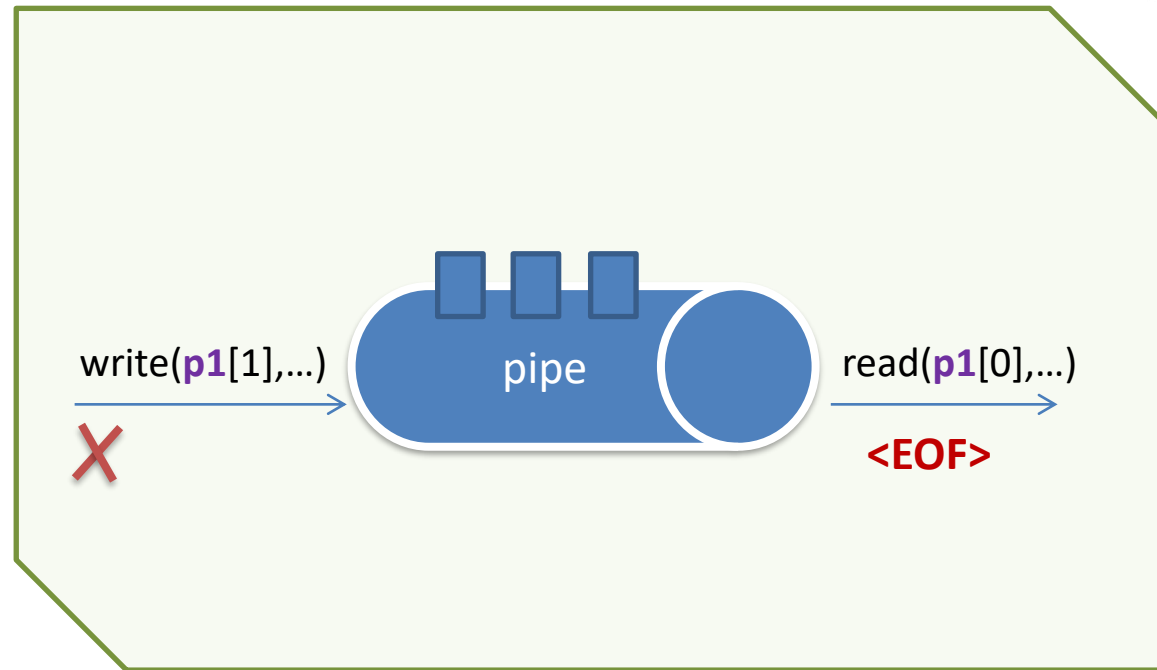
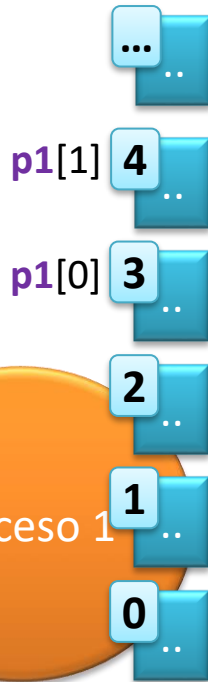
```
    close(p1[1]);
```

```
    close(p1[0]);
```

```
    close(p1[0]);
```

```
    ...
```

```
}
```



- Cuando todos los posibles procesos escritores en el pipe cierran la parte de escritura, entonces se manda un final de fichero (EOF) a los lectores.

Descriptores de ficheros

2.- fork()

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2];
```

```
...
```

```
pipe(p1);
```

```
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

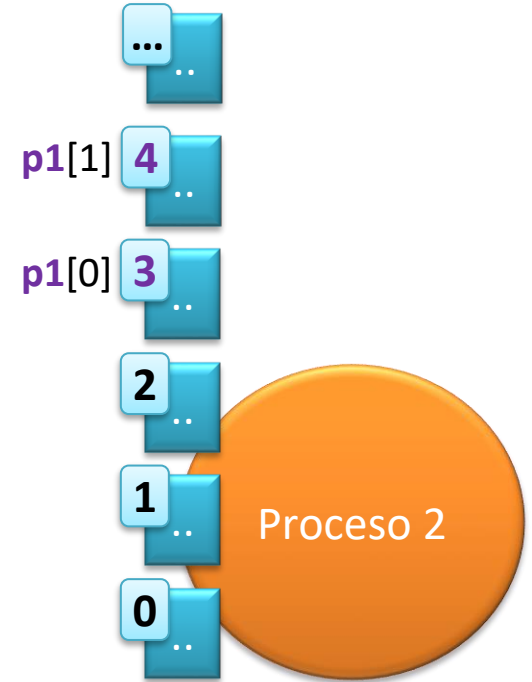
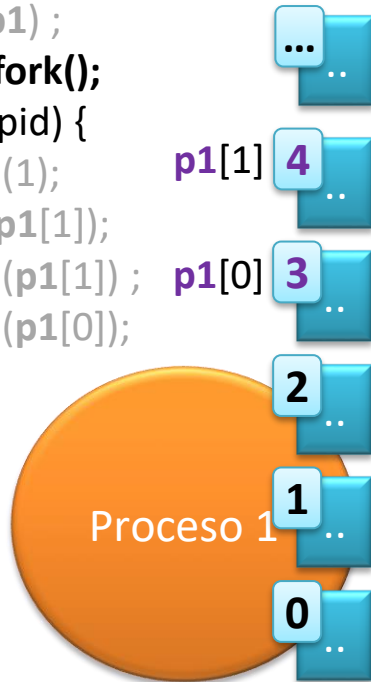
```
    dup(p1[1]);
```

```
    close(p1[1]);
```

```
    close(p1[0]);
```

```
    ...
```

```
}
```



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

Descriptores de ficheros

2.- fork()

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2] ;
```

```
...
```

```
pipe(p1) ;
```

```
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

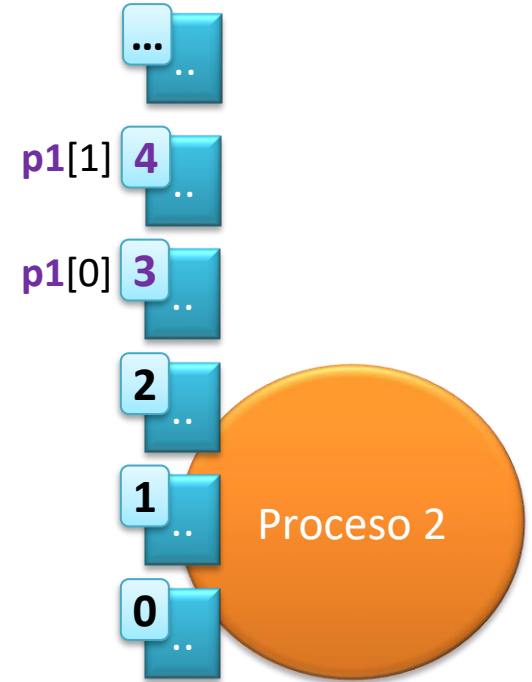
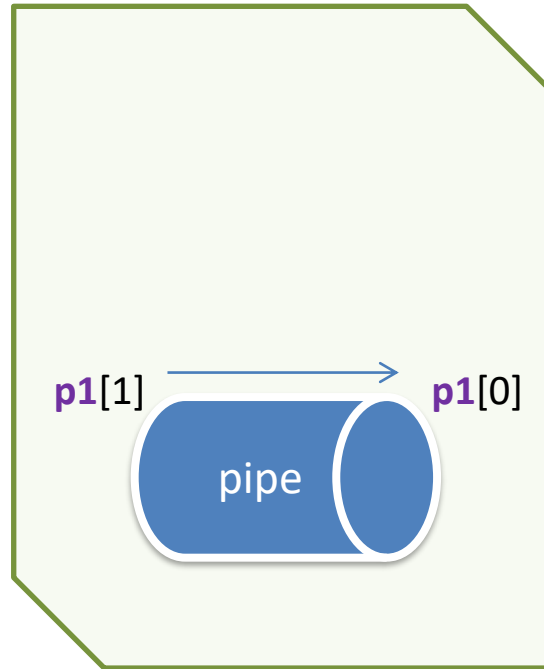
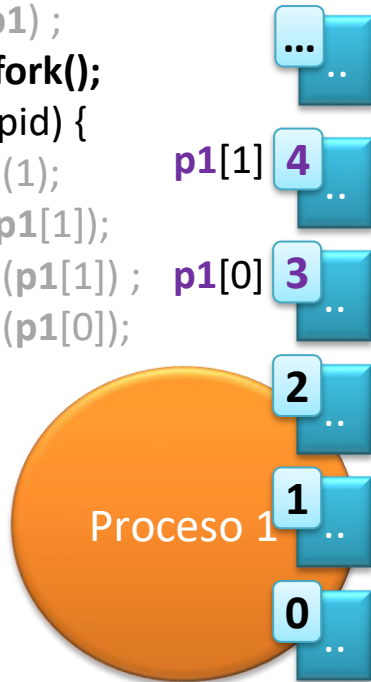
```
    dup(p1[1]);
```

```
    close(p1[1]) ;
```

```
    close(p1[0]);
```

```
...
```

```
}
```



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

Descriptores de ficheros

3.- redirección

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2];
```

```
...
```

```
pipe(p1);  
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

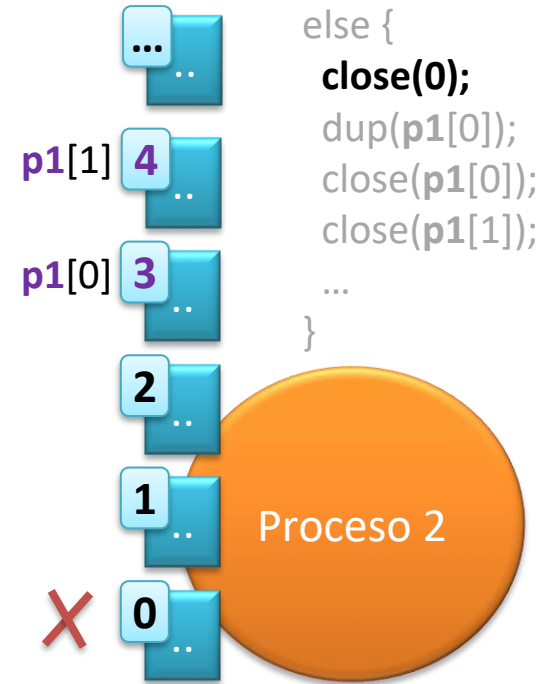
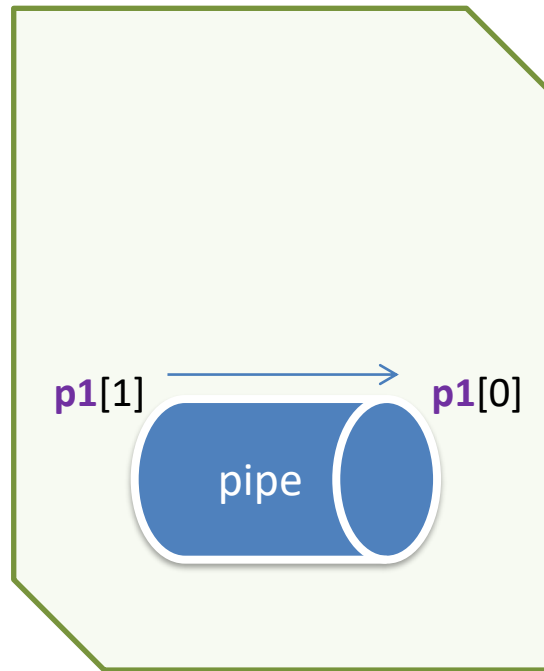
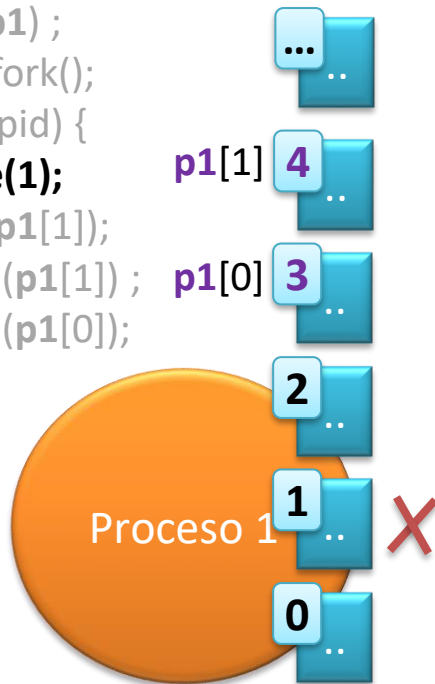
```
    dup(p1[1]);
```

```
    close(p1[1]);
```

```
    close(p1[0]);
```

```
    ...
```

```
}
```



```
else {
```

```
    close(0);
```

```
    dup(p1[0]);
```

```
    close(p1[0]);
```

```
    close(p1[1]);
```

```
    ...
```

```
}
```

Redirección de la salida estándar en el padre...

Redirección de la entrada estándar en el hijo...

Descriptores de ficheros

3.- redirección

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2] ;
```

```
...
```

```
pipe(p1) ;  
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

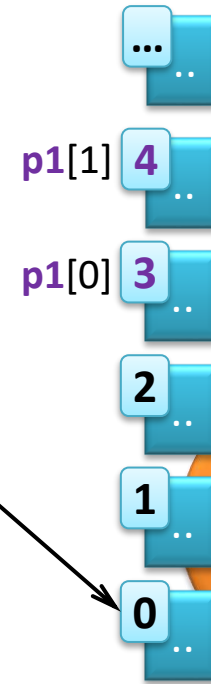
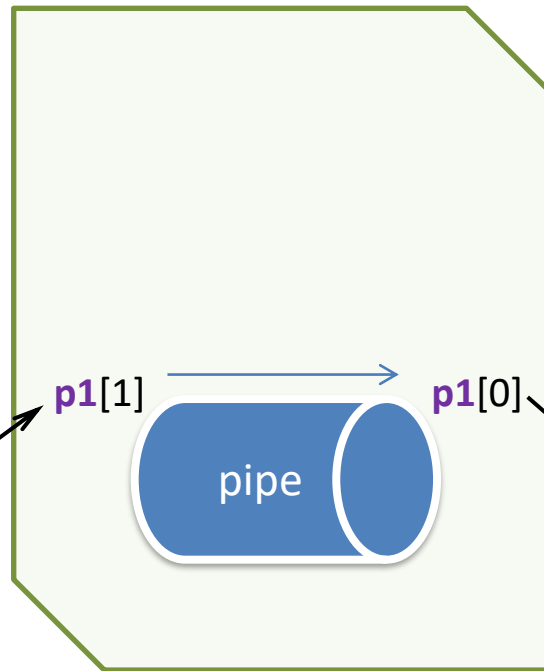
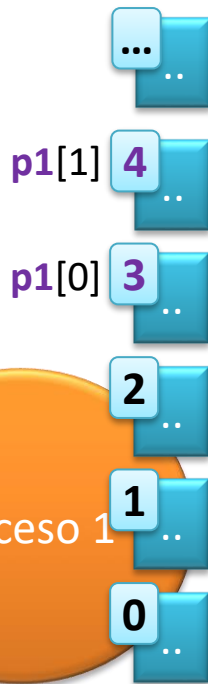
```
    dup(p1[1]);
```

```
    close(p1[1]) ;
```

```
    close(p1[0]);
```

```
    ...
```

```
}
```



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

Redirección de la salida estándar en el padre...

Redirección de la entrada estándar en el hijo...

Descriptores de ficheros

4.- limpieza

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2];
```

```
...
```

```
pipe(p1);
```

```
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

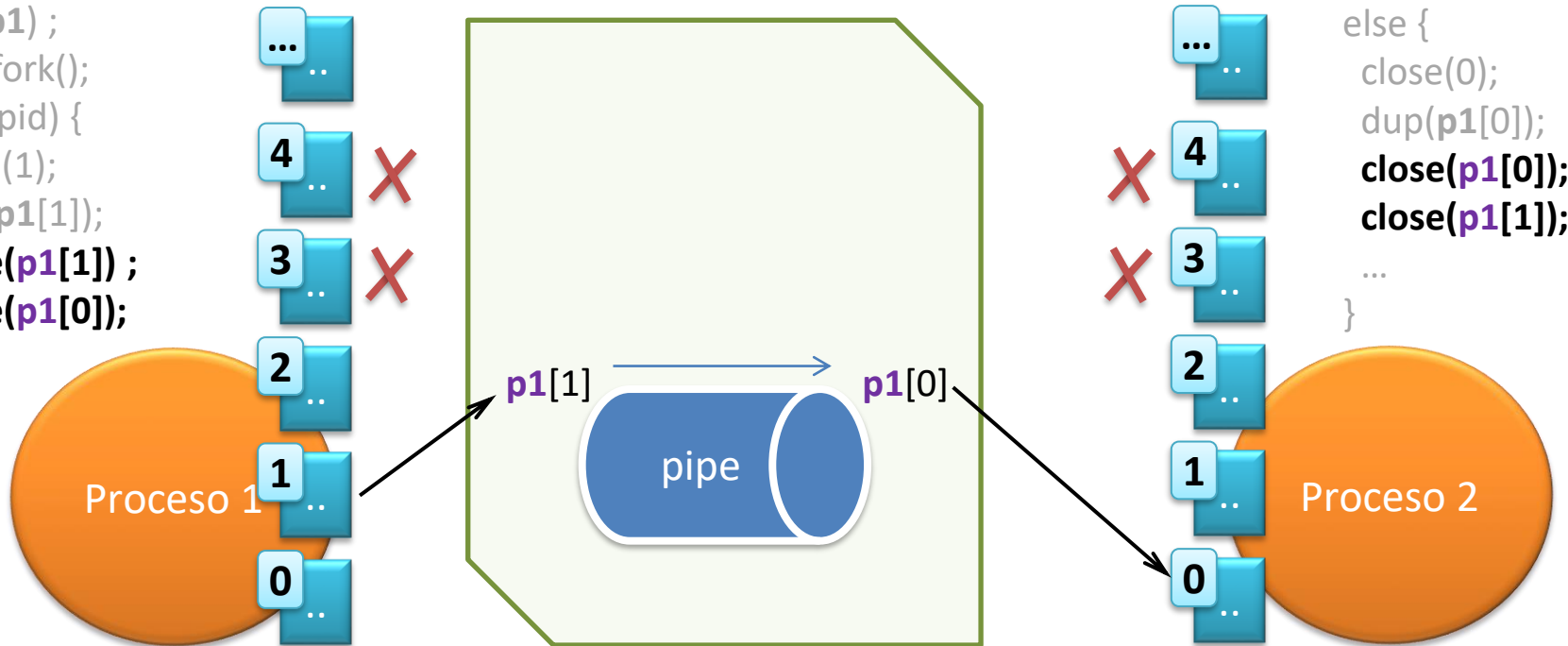
```
    dup(p1[1]);
```

```
    close(p1[1]);
```

```
    close(p1[0]);
```

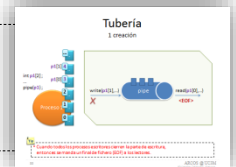
```
...
```

```
}
```



Cierre de los descriptores que no se usan en el padre...

Cierre de los descriptores que no se usan en el hijo...



Descriptores de ficheros

resumen

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
int p1[2];
```

```
...
```

```
pipe(p1);
```

```
pid = fork();
```

```
if (0!=pid) {
```

```
    close(1);
```

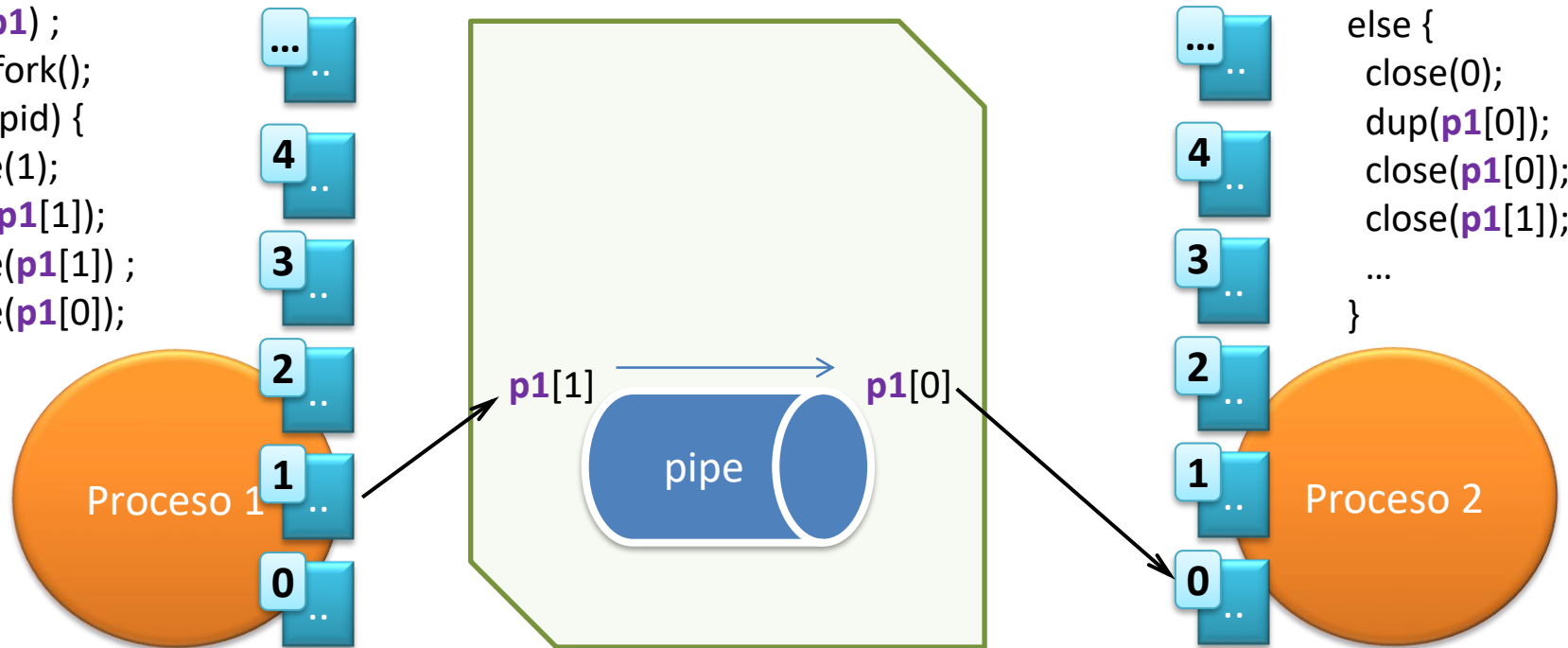
```
    dup(p1[1]);
```

```
    close(p1[1]);
```

```
    close(p1[0]);
```

```
...
```

```
}
```



Descriptores de ficheros

resumen

- Linux: redirección y tuberías
- Los descriptores de ficheros
 - Redirección
 - Duplicación
 - `fork()`
- Tuberías

```
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

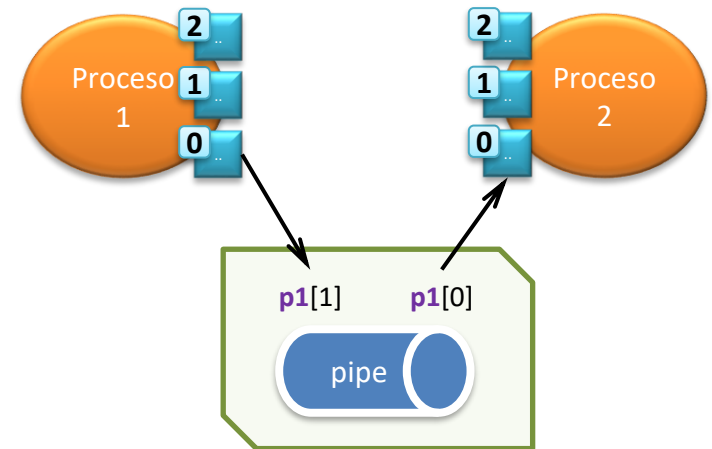
2) `fork()`

3) Redirección (padre)

4) Limpieza (padre)

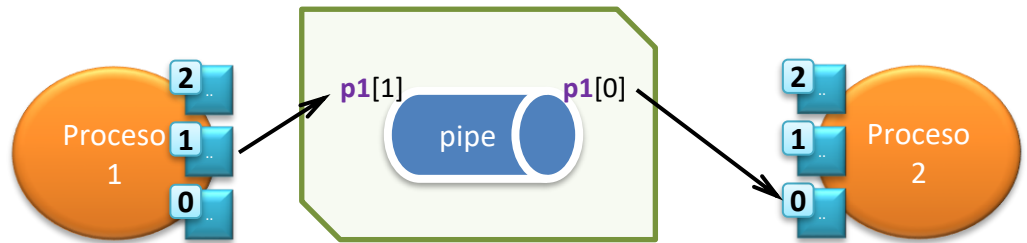
3) Redirección (hijo)

4) Limpieza (hijo)



Tuberías

limitaciones



- **Semi-duplex:**
 - En un sentido: los datos son escritos por un proceso en un extremo de la tubería y leídos por otro proceso desde el otro extremo del mismo.
- Solo se pueden utilizar entre **procesos emparentados**, que tengan un ancestro en común.
- La **lectura** es **destructiva**.

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;
}
```


Lección 3

Señales, excepciones y pipes

Sistemas Operativos
Ingeniería Informática