P3: Redirection

Ion Mandoiu
Laurent Michel
Revised by M. Khan and J. Shi

Review

- Function open() returns a file descriptor, a non-negative integer
- The file descriptor is used later on in functions like read() and close()
- Every opened file has a file descriptor
  - stdin: 0, stdout: 1, stderr: 2

- Files opened in a process remain open after fork() and exec

Shell redirections

Available when executing commands in your shell (e.g. bash)

- **Implemented with the close/open/dup technique**

```
$ command < infile  > outfile
```

<	infile	: Take input from file *infile*
>	outfile	: Send output to file *outfile*

- Other variants

>>	outfile	: Append output to file *outfile*
2>	outfile	: Send errors to file *outfile*
&>	outfile	: Send both output and errors to file *outfile*

Read the manual for more variants like 2>>, 2>&1, etc.

Shell redirection examples

```
$ sort < file.txt  > sorted.txt
```

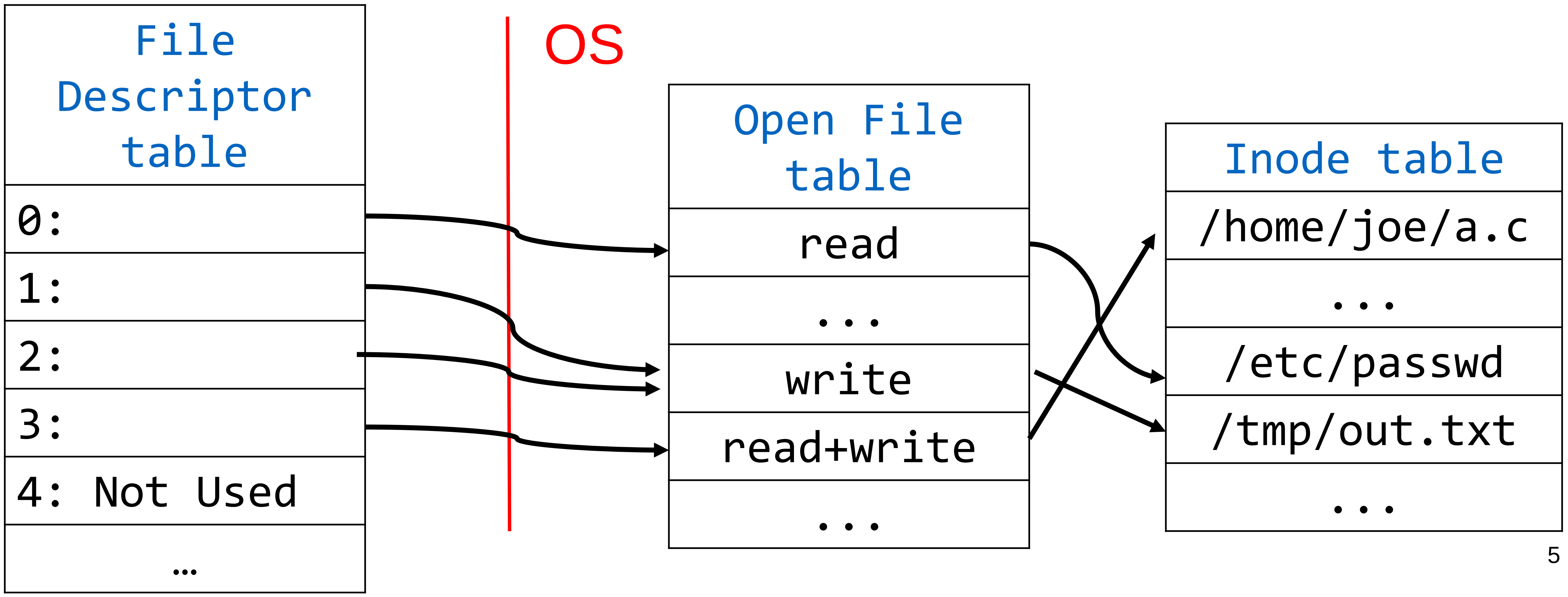
- sort will read lines from file.txt, instead of terminal
- The output of sort will be saved in sorted.txt
  - You cannot see it on screen

The statements in sort are not changed.

They read from stdin (0), and print to stdout (1)

File descriptor table

- Each process has a **file descriptor table**
  - Holds indices of entries into the **Open File Table** managed by OS
- The system-wide **Open File Table**
  - Records the *mode* of the opened files (e.g., reading, writing, appending)
  - Holds index into **the Inode Table** that has the actual file name and location on disk



Duplicating File Descriptors

- Do not change file descriptor table directly
- Used open() and close() and two new functions

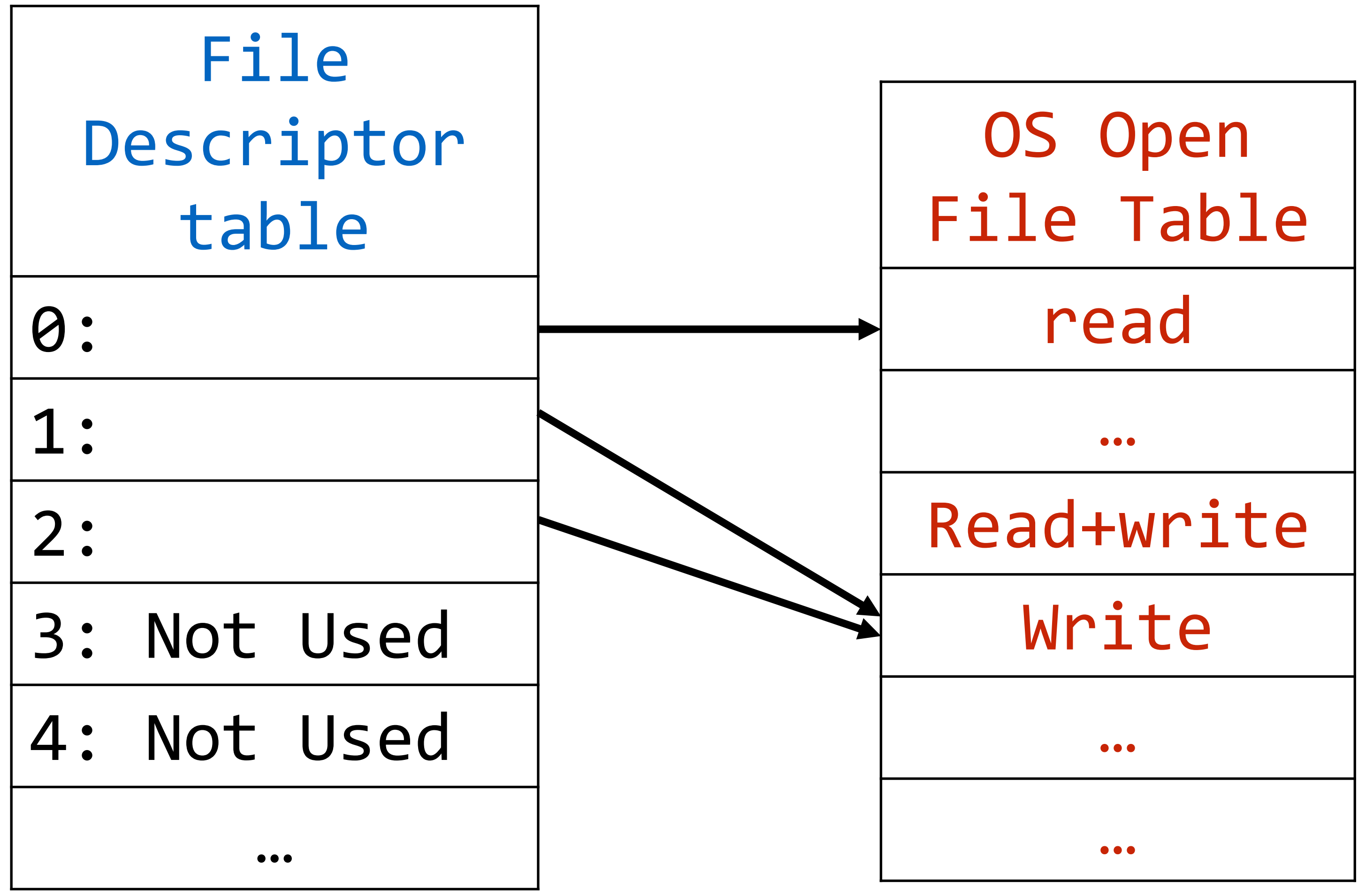
```
#include <unistd.h>

int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

- dup() copies oldfd to the **first available entry** (in FD table)
- dup2() copies oldfd to newfd
  - Closes newfd first if it is in use

There is dup3(), but it is not in POSIX. We should not use it in this course.

Example: stdout redirect

- A program can change its standard output
- How?

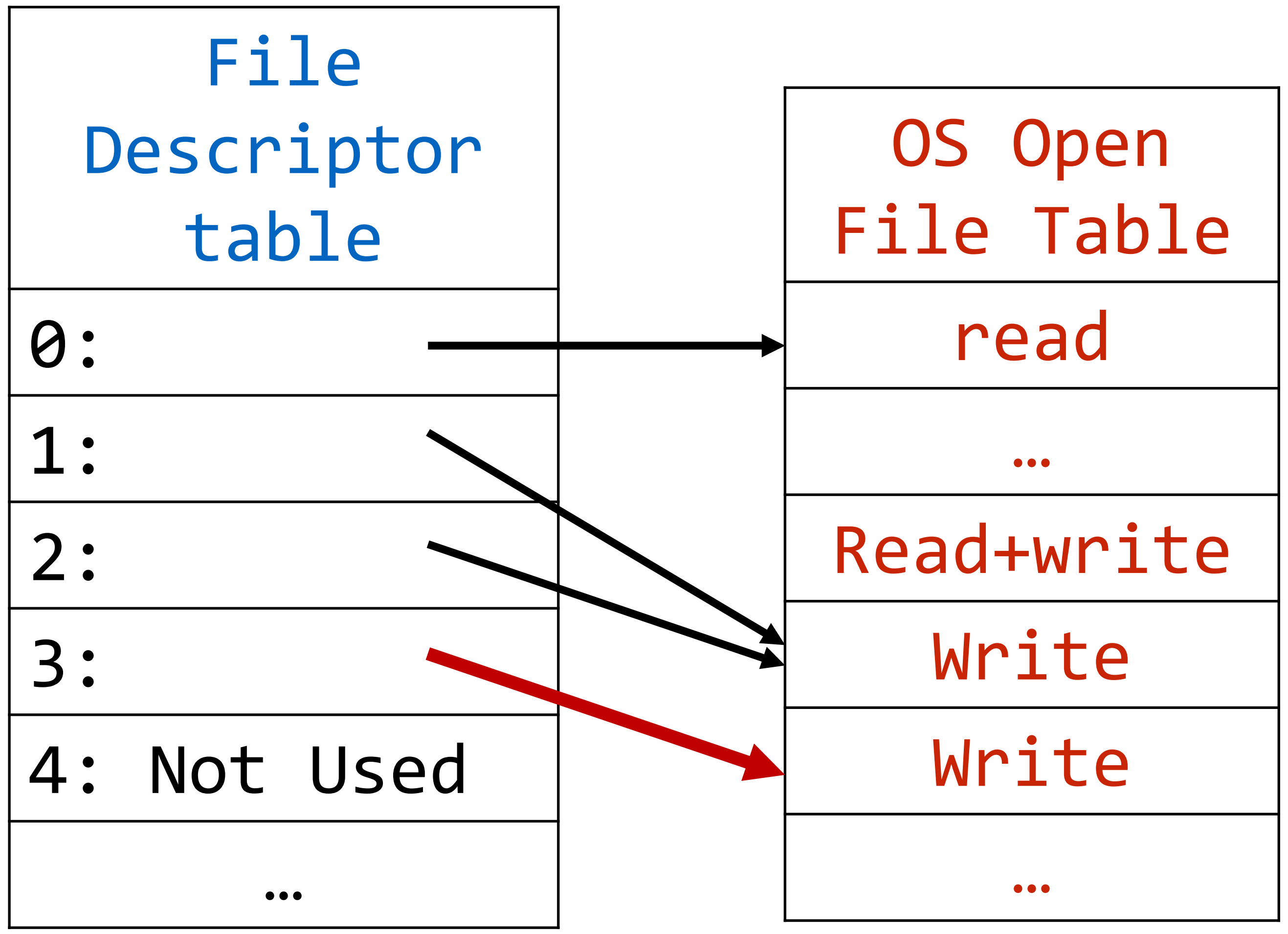


Steps for redirecting stdout

1. open(). open a file (and save the file descriptor in fd)
2. dup2(). copy fd to 1 (so that the file descriptor 1 points to the file just opened)
3. close(fd)

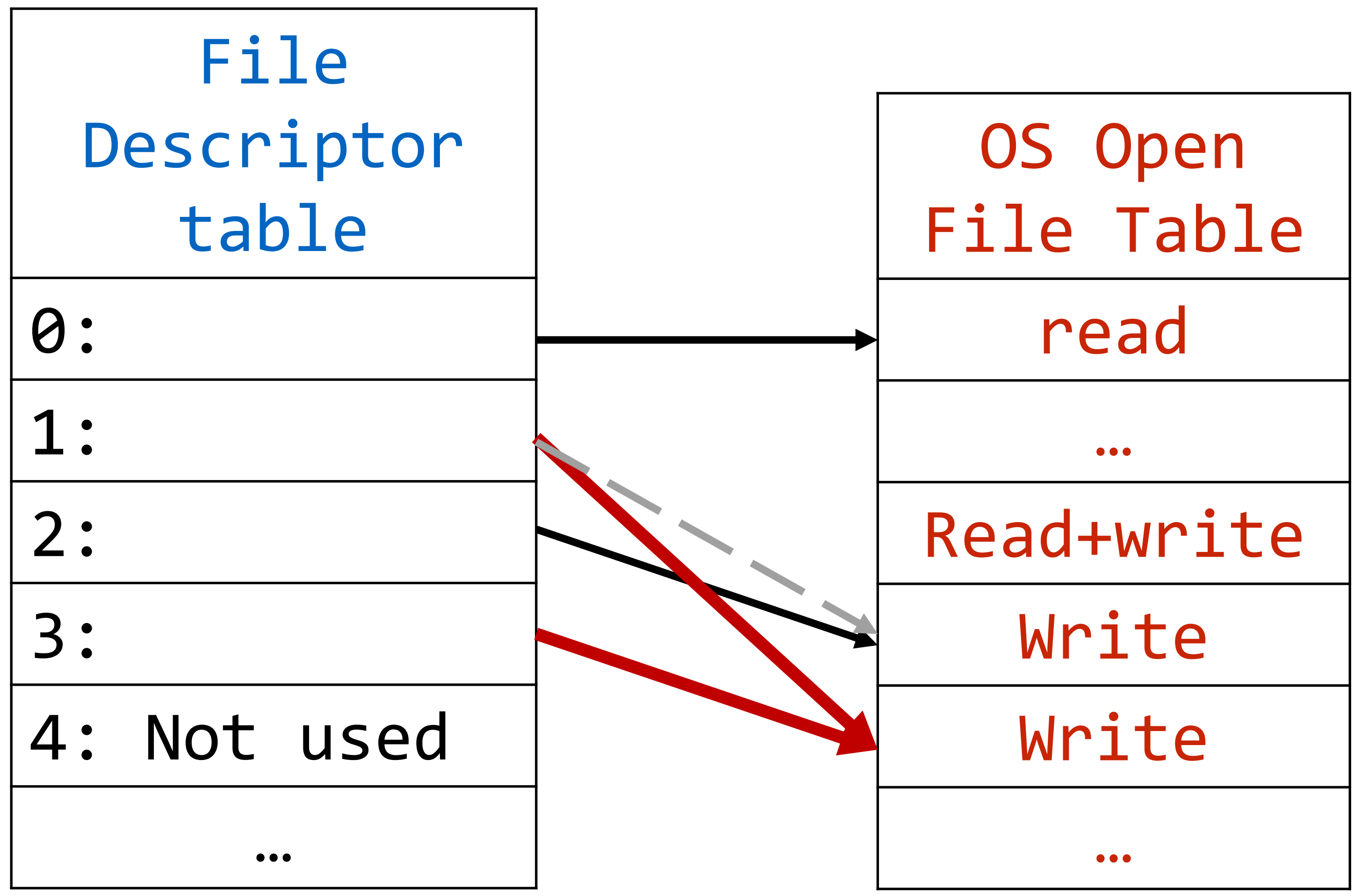
Example: stdout redirect

- Open the new file for writing; 3 is the returned fd
  - We will use 3 instead of a variable in this example



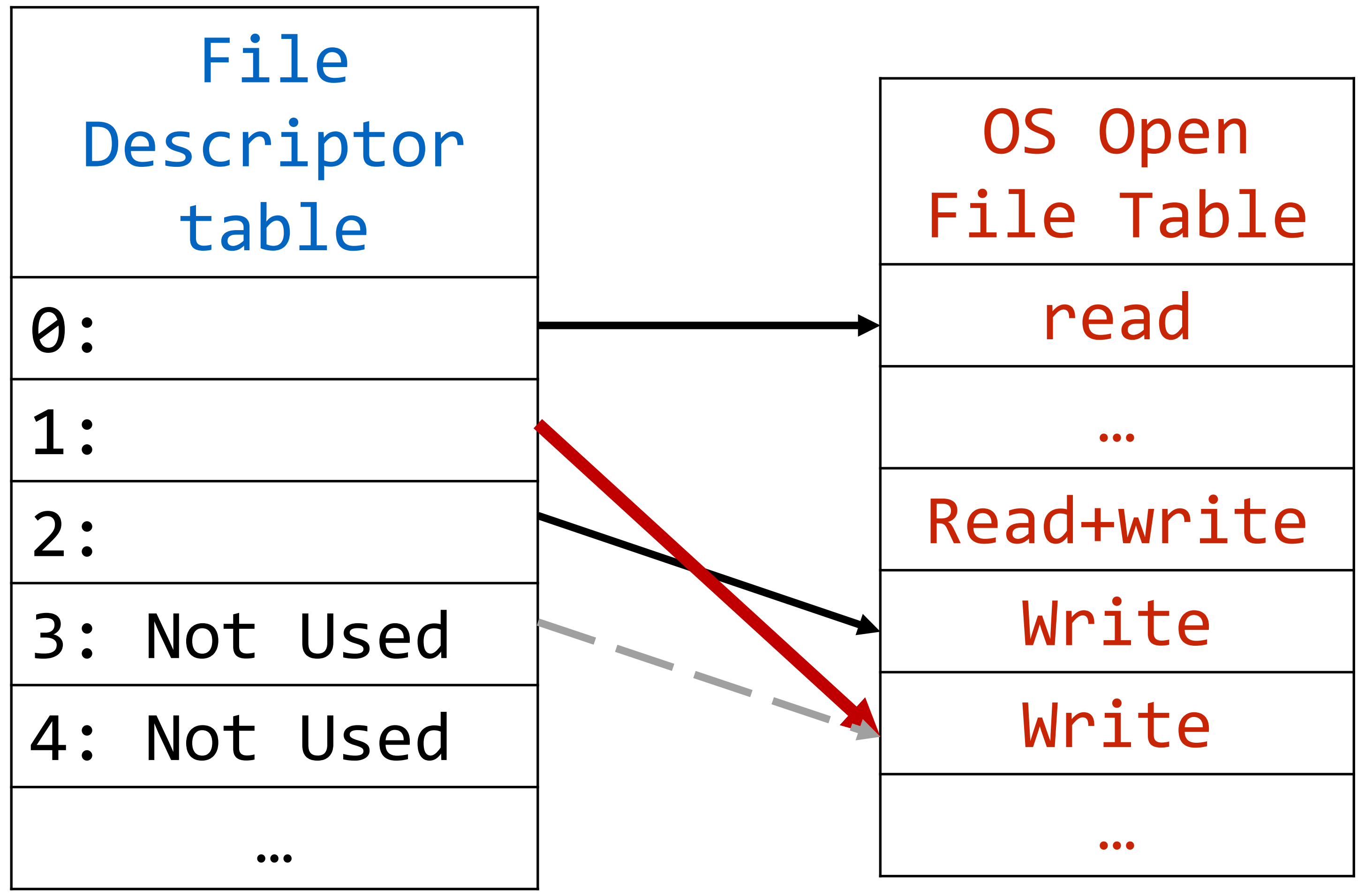
Example: stdout redirect

```
// Method 1: two functions. not atomic
close(1);      dup(3);
// Method 2: better. dup2() closes newfd first
dup2(3, 1);
```



Example: stdout redirect

```
close(3);
```



Implementing redirections

How does bash do redirection for other processes?

When bash starts a new process, it does fork() and exec

Recall that the file descriptor table **is preserved during fork & exec**

Idea:

In child process, set up proper file descriptors before upgrading

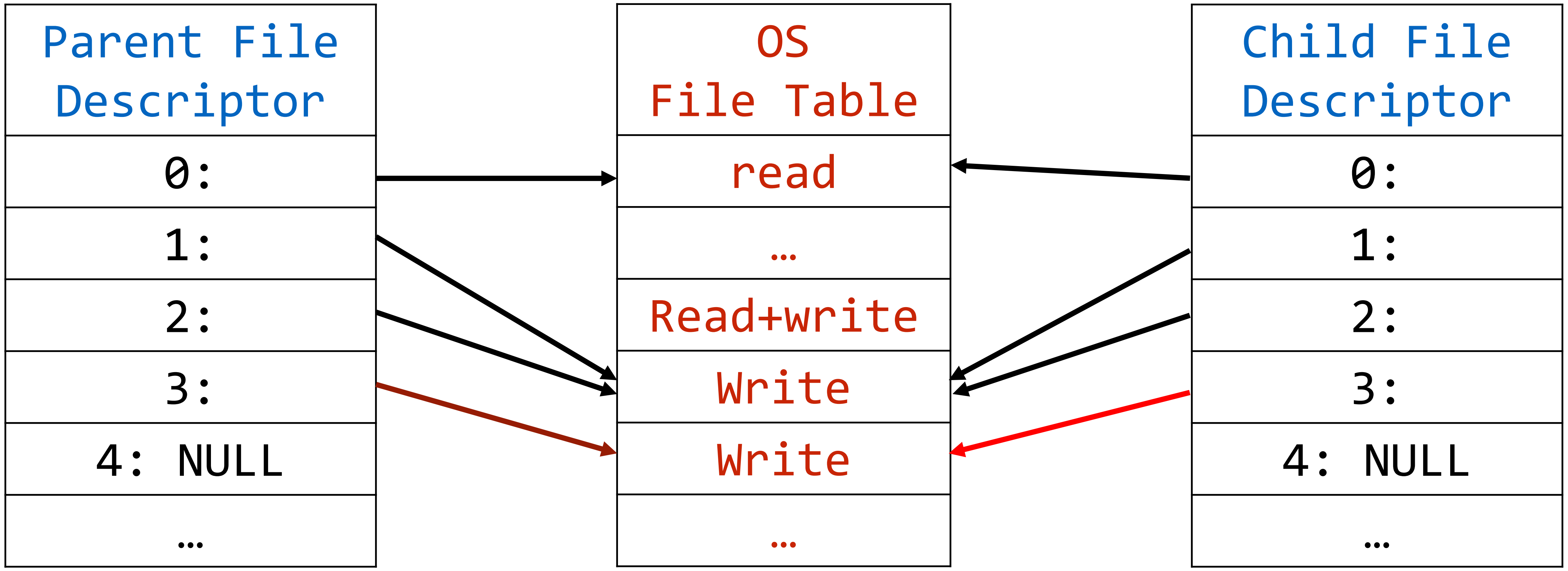
- Simply *change* the files corresponding to stdin, stdout, or stderr

Example: redirecting stderr for another program

Assume the parent uses FD 3.

After fork(), the child has the same file descriptors as the parent

Close FDs that are not needed !



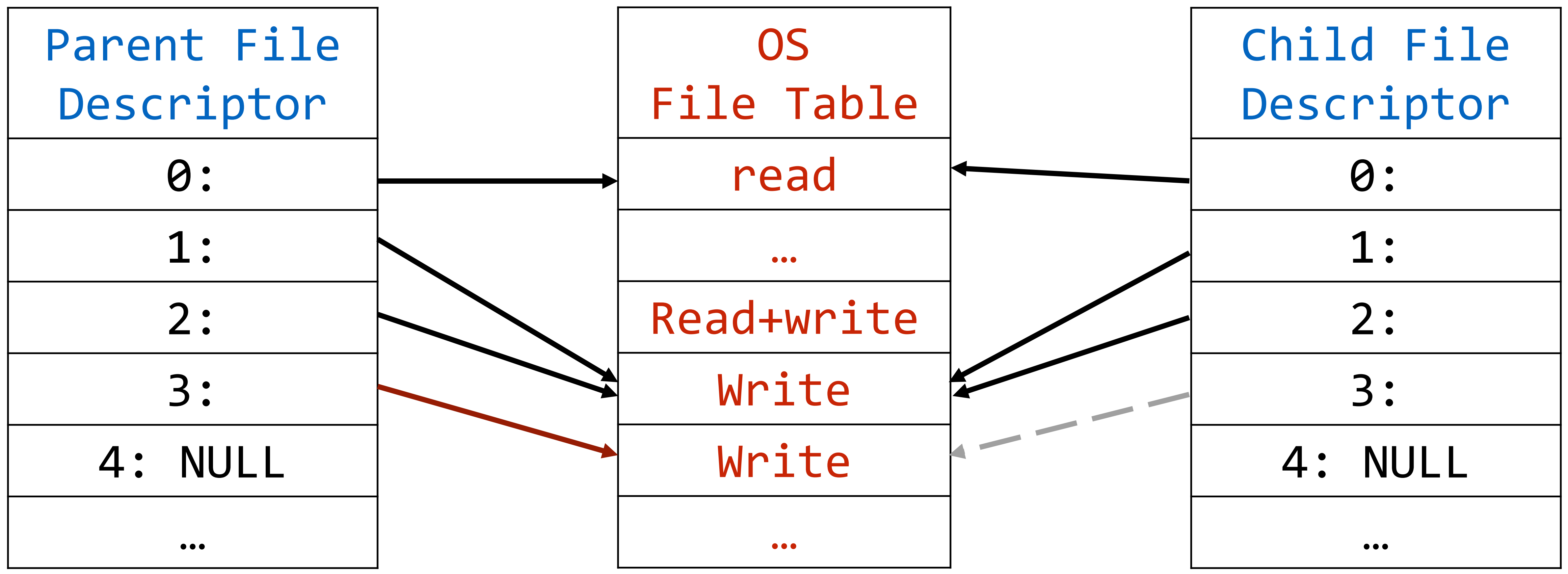
Example: redirecting stderr for another program

Assume the parent uses FD 3.

After fork(), the child has the same file descriptors as the parent

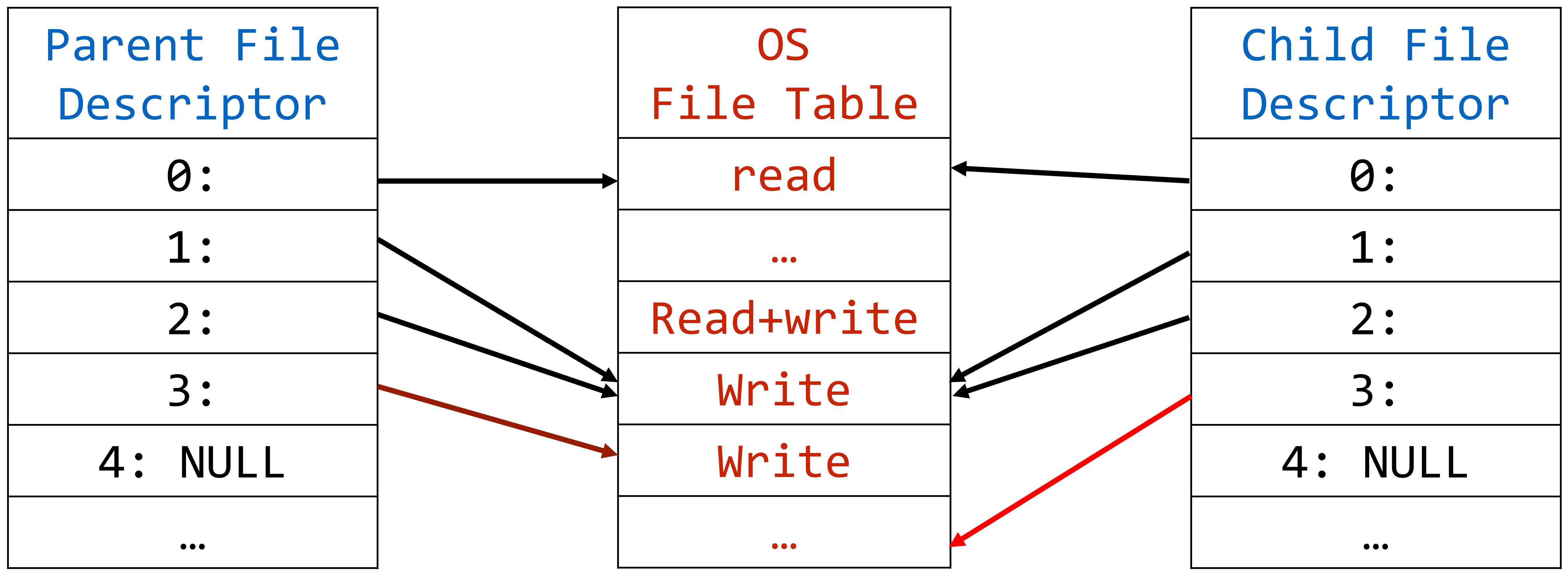
  Close FDs that are not needed !

How do you close FD 3 in child process?



Example: redirecting stderr for another program

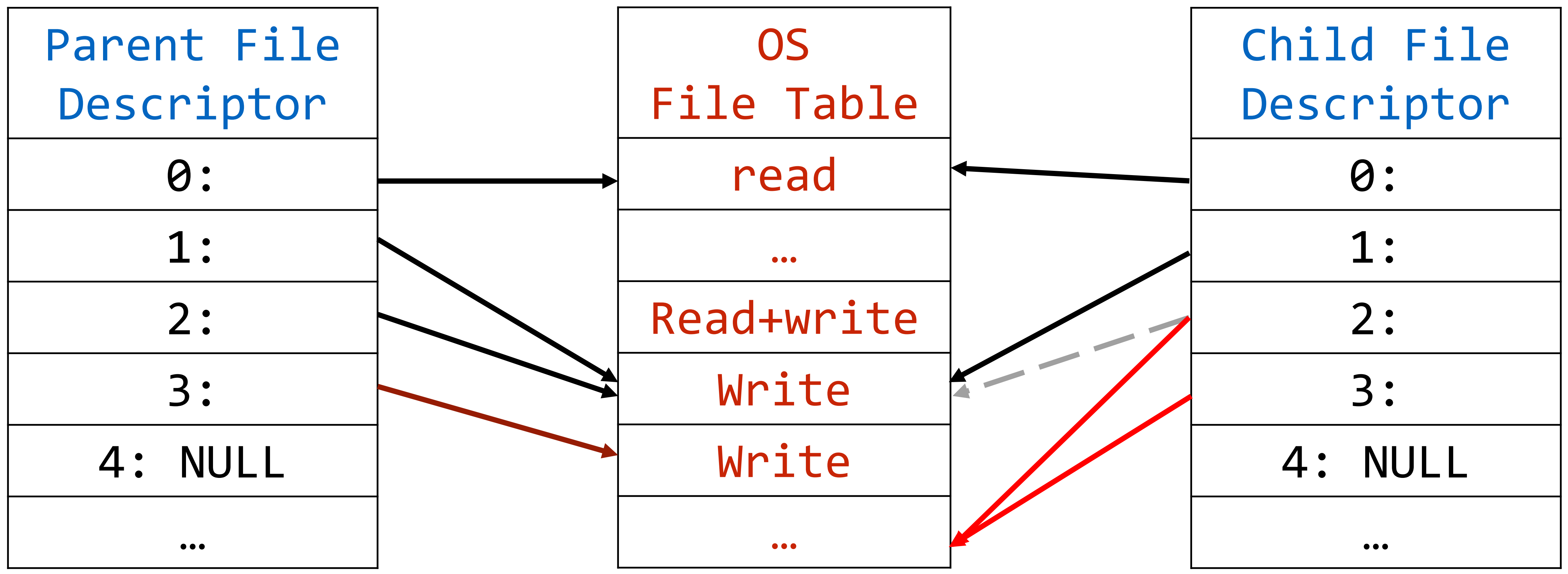
- Child: open the file (to save error output)



Example: redirecting stderr for another program

- Child:

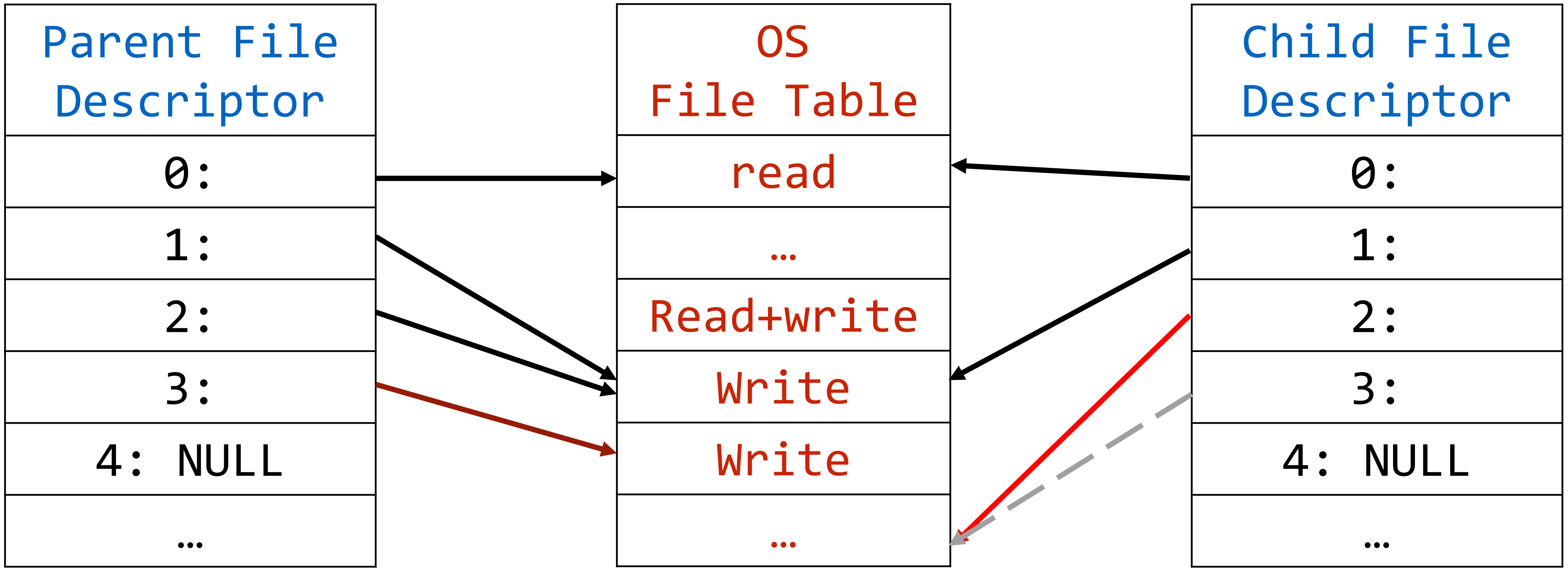
```
dup2(2, 3);
close(3);
```



Example: redirecting stderr for another program

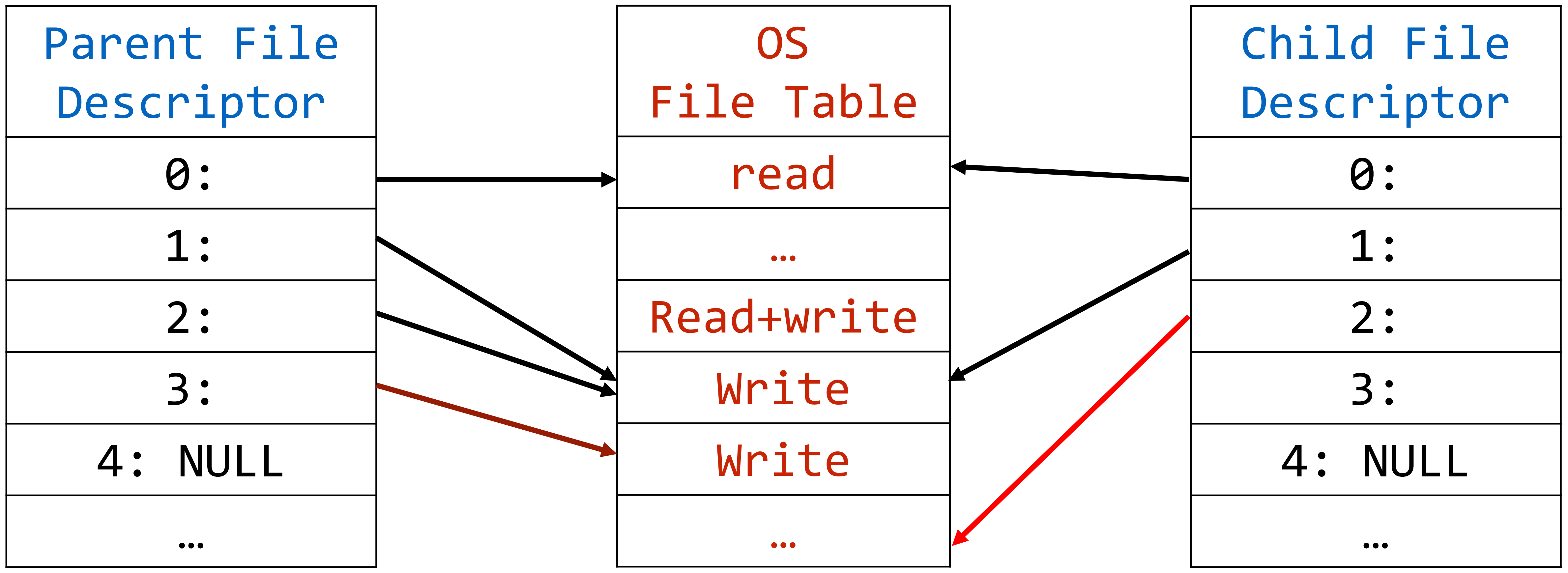
- Child:

```
dup2(2, 3);
close(3);
```



Example: redirecting stderr for another program

- Child: exec and the new executable has redirected stderr!
- On exec, open file descriptors are preserved !!!



Summary of Steps in Child Process

1. Close FDs that are opened in parent and not needed in child
2. open(). Open a file (and save the file descriptor in fd)
3. dup2(). Copy FD to the right place
4. close(fd)
5. Exec

Redirecting stdout for a child process

A process would like to start a new program, and redirect stdout of the new process to a file.

Where & when should the file be opened?

Select the best option.

A. Before fork(), in parent
B. After fork() in parent
C. Before exec in child (after fork())
D. After exec in child (after fork())
E. None of the above

More question on redirecting stdout

- Where & when should the new file be opened ?
- Where & when should you call close(1) ?
- Where & when should you call dup ?
- Where & when should close(newfd) be called ?