CS 3305A

Processes

Lecture 5

Sept 23 2019

Pipes

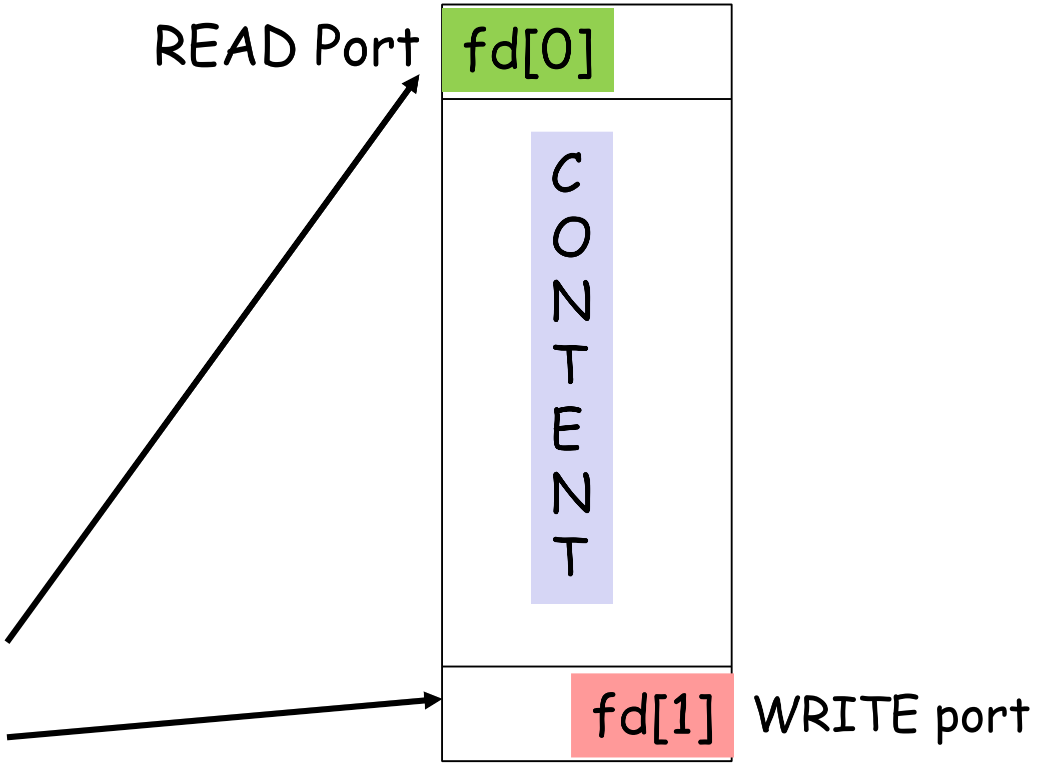
- The pipe function can be used to provide the shared memory to allow communication between two processes
- We will first provide a general discussion of the pipe function which is to be followed by a discussion of how it applies to communicate between parent and child process
    - pipe() before fork()
    - pipe() after fork()
    - Single R/W operations by parent and child
    - Multiple R/W operations by parent and child

Creating a Pipe

```
int fd[2];
int status;

status = pipe(fd);
```

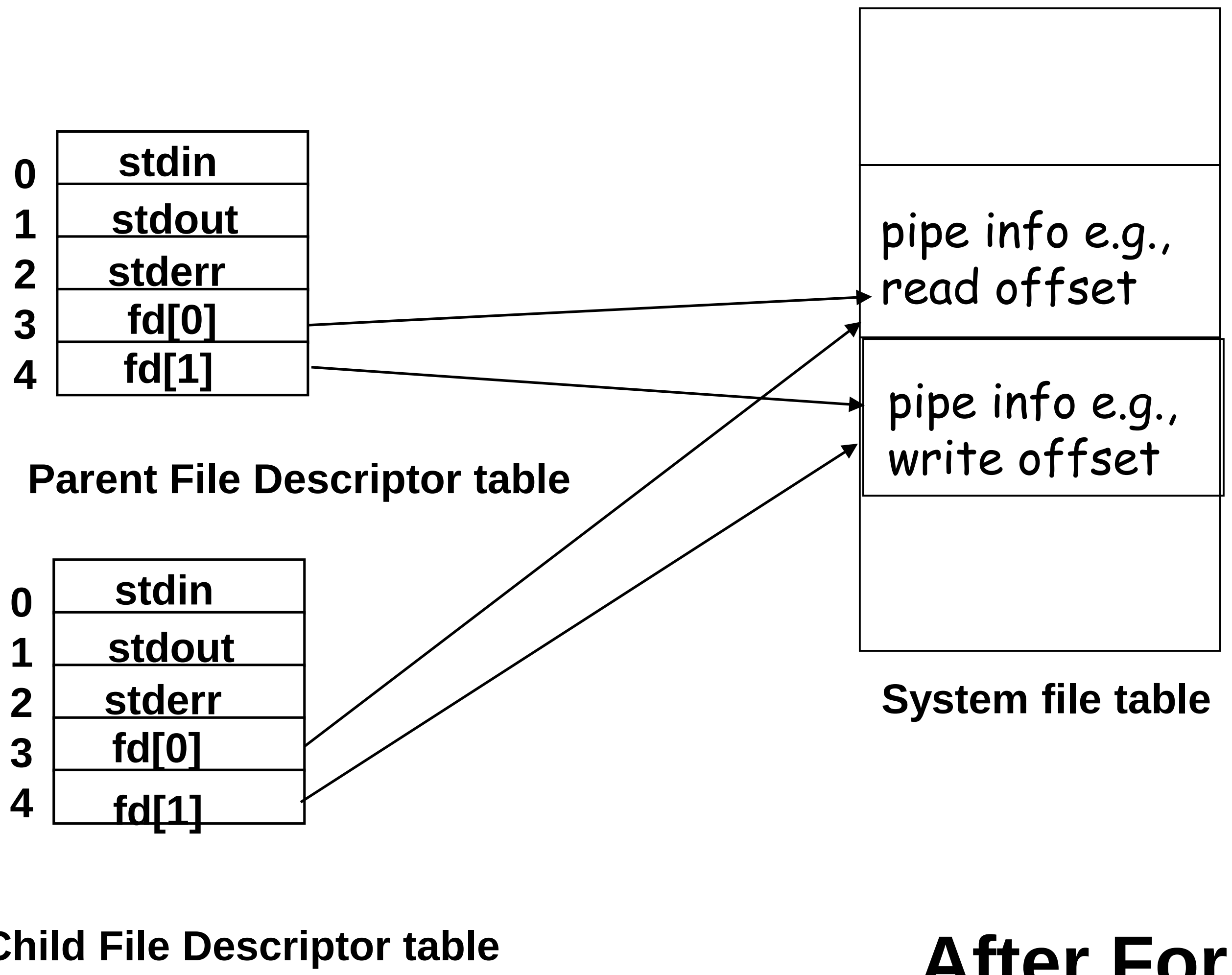
- Returns 0 if ok, -1 on error
- Attached two file descriptors
  - fd[0] is open for reading
  - fd[1] is open for writing



Fork and Pipes

- A fork copies the file descriptor table to the child
- fd[0] of parent and child points to the same location in the pipe.
- fd[1] of parent and child points to the same location in the pipe.

Fork and Pipes


0
1
2
3
4
stdin
stdout
stderr
fd[0]
fd[1]
Parent File Descriptor table
0
1
2
3
4
stdin
stdout
stderr
fd[0]
fd[1]
Child File Descriptor table
pipe info e.g., read offset
pipe info e.g., write offset
System file table

After Fork

Pipes

- When the pipe is full: By default, if a writing process attempts to write to a full pipe, the system will automatically block the process until the pipe is able to receive the data
  - The OS has a limit on the buffer space used by the pipe and if you hit the limit, write will be blocked

- When the pipe is empty: if a read is attempted on an empty pipe, the process will block until data is available

Example

- pipe_SRW.c
- pipe_MRW.c