

CS 3305A

Process

Lecture 5

Sept 25th 2023

Copyright Notice:

1. Lecture slides are being provided with permission from the copyright for in-class (CS3305A) use only. The slides must not be reproduced or provided to anyone outside of the class.
2. All download copies of the slides and/or lecture recordings are for personal use only.
3. Students must destroy these copies within 30 days after receipt of final course evaluations

Pipe()

- ❑ Values of the variables from parent get copied to child but they reside in different memory locations

```
Int main()  
{  
    int_pid pid;  
    int x = 10, y = 20;  
    pid = fork();
```

```
    if (pid > 0)  
    {  
        wait(NULL);  
        Printf(" x = %d y = %d", x, y);  
    }
```

x = 10 y = 20

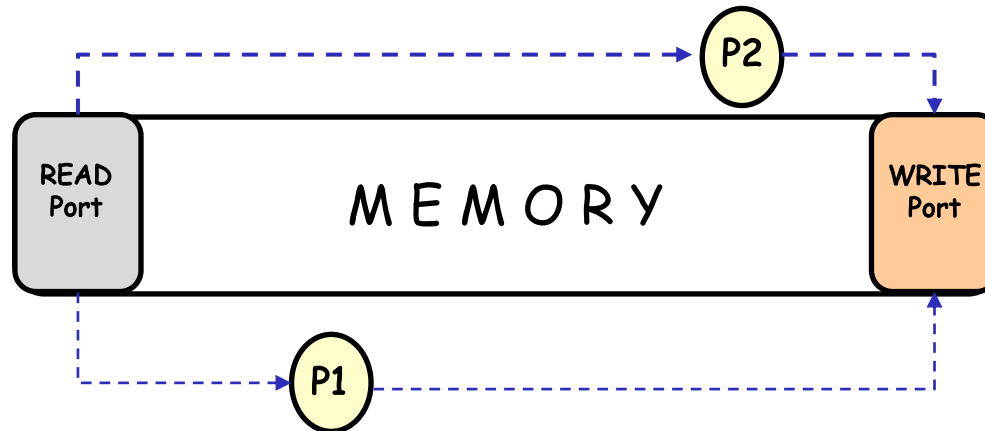
```
    if (pid == 0)  
    {  
        x = 100;  
        y = 200;  
        Printf("\n x = %d y = %d", x, y);  
    }  
    return 0;  
}
```

x = 100 y = 200

How parent and child will communicate?

Pipe()

- ❑ The `pipe()` function can be used to provide the shared memory to allow communication between two processes



- ❑ We will discuss how `pipe()` can be used to communicate between parent and child process
 - ❑ `pipe()` must be created before `fork()`
 - ❑ Single R/W operations by parent and child
 - ❑ Multiple R/W operations by parent and child

Creating a Pipe

```
int port[2];
```

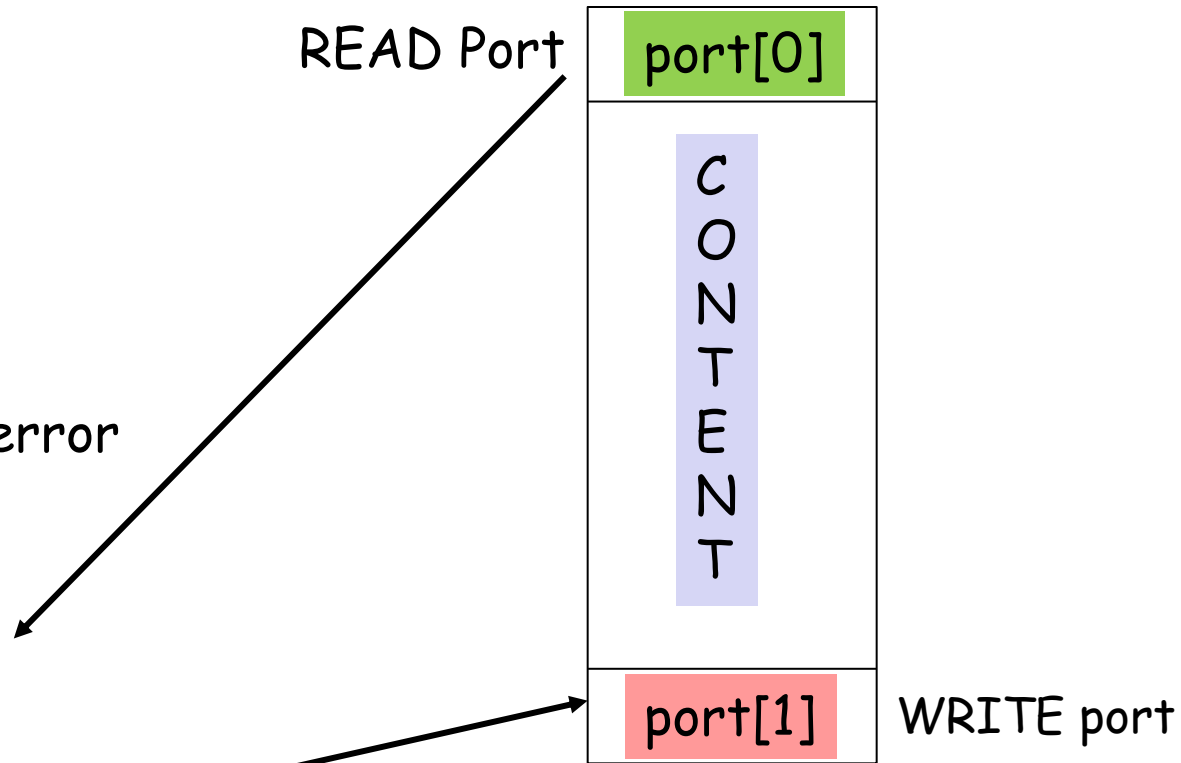
```
int status;
```

```
status = pipe(port);
```

- ❑ Returns 0 if ok, -1 on error
- ❑ Attached two file descriptors

- `port[0]` is open for reading

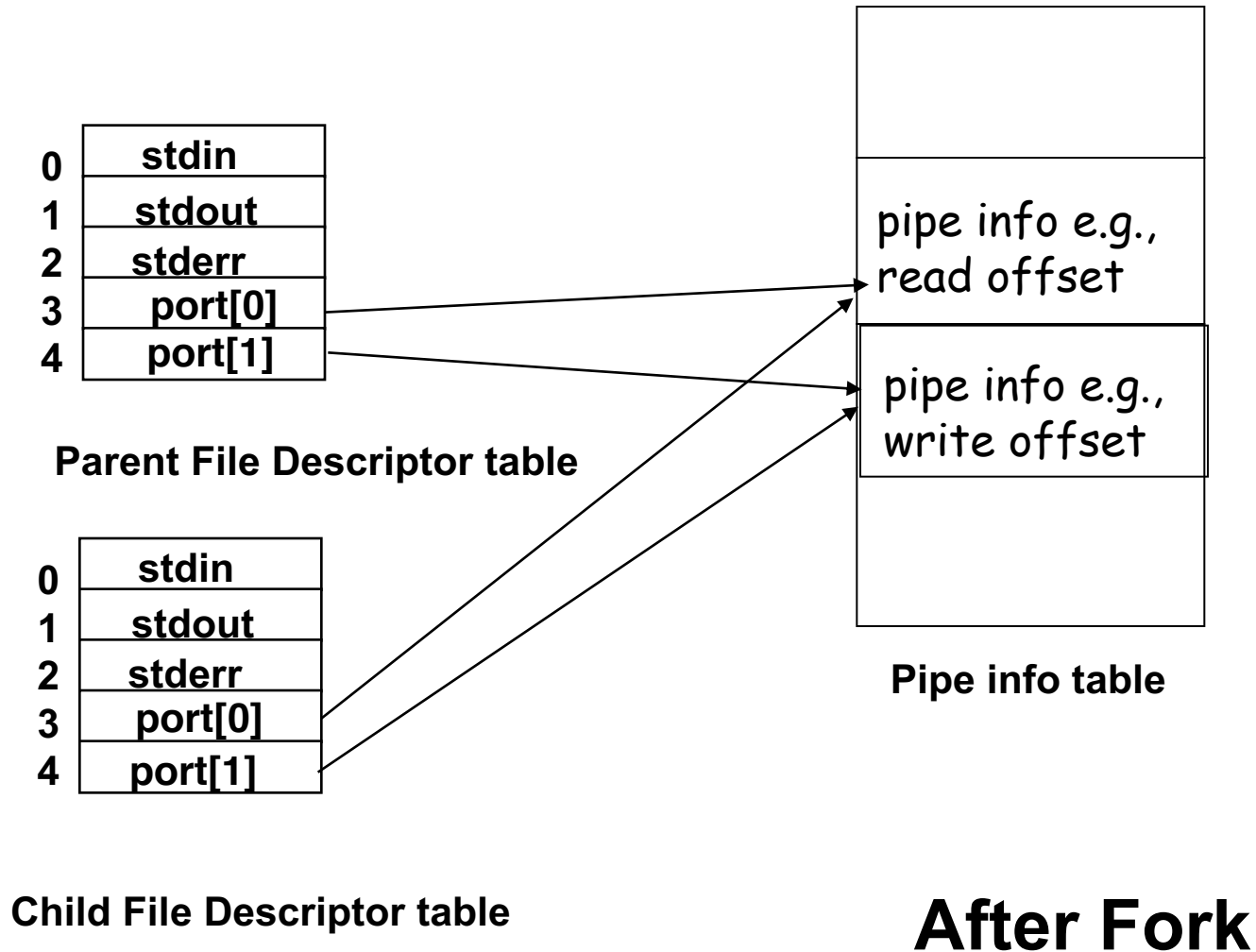
- `port[1]` is open for writing



Fork and Pipes

- ❑ A `fork()` copies the port info to the child
- ❑ `port[0]` of parent and child points to the same location in the pipe.
- ❑ `port[1]` of parent and child points to the same location in the pipe.

Fork and Pipes



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