Operating Systems: Lab 3

1/20/21

Juliana Shihadeh

This week’s focus in lab:

1. To develop multi-process application programs
2. To demonstrate the use of pipes as an inter-process communication (IPC) mechanism
3. To demonstrate the use of pthreads

What do processes and threads allow us to do?

Multi-processing and Multi-threading

To maintain the correct synchronization

To be able to use Parallelism and run multiple things at the same time

Reminder about processes

- Multiple processes can be running the same program, with each process having its own copy of the program
  - Each process has its own address space
  - Each process's address space holds its own copy of the program
- If you copy a process, its memory is copied in the state it is
  - Once its copied, what happens in the new process does not affect the original process
  - The state of memory it copies also includes any files that were open in the other process
- They do not share memory like threads can
  - Thus, in order to share information with each other they uses one of the IPC - Inter-Process Communication - Mechanisms
    - 3 IPC's provided by kernels in total: Pipes, Shared Memory, and Message Queues

Shared Memory

One Process creates a shared segment using:

```
id = shmget( key, MSIZ, IPC_CREAT | 0666);
```

Other processes get the shared segment's id using:

```
id = shmget( key, MSIZ, 0 )
```

Processes attach themselves to the shared memory segment to communicate with each other using:

```
ctrl = (struct info *) shmat( id, 0, 0);
```

And when processes are done, they just detach themselves from the shared memory segment

Message Queues

Process creates message queue to use:

```
id = msgget( key, IPC_CREAT | 0644);
```

Other processes get the id to the shared message queue using:

```
id = shmget( key,0 )
```

Processes send a message to the queue using:

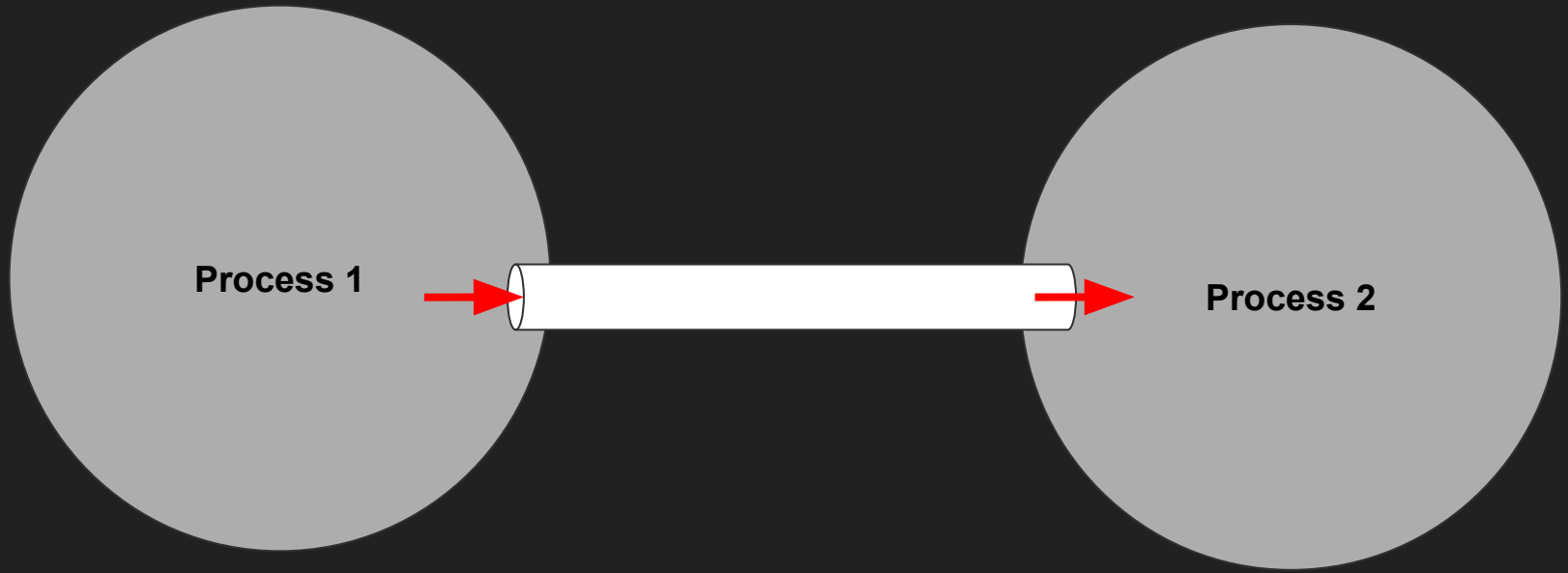
```
v = msgsnd( id, ptr, length,IPC_NOWAIT);
```

Processes receive a message from the queue using:

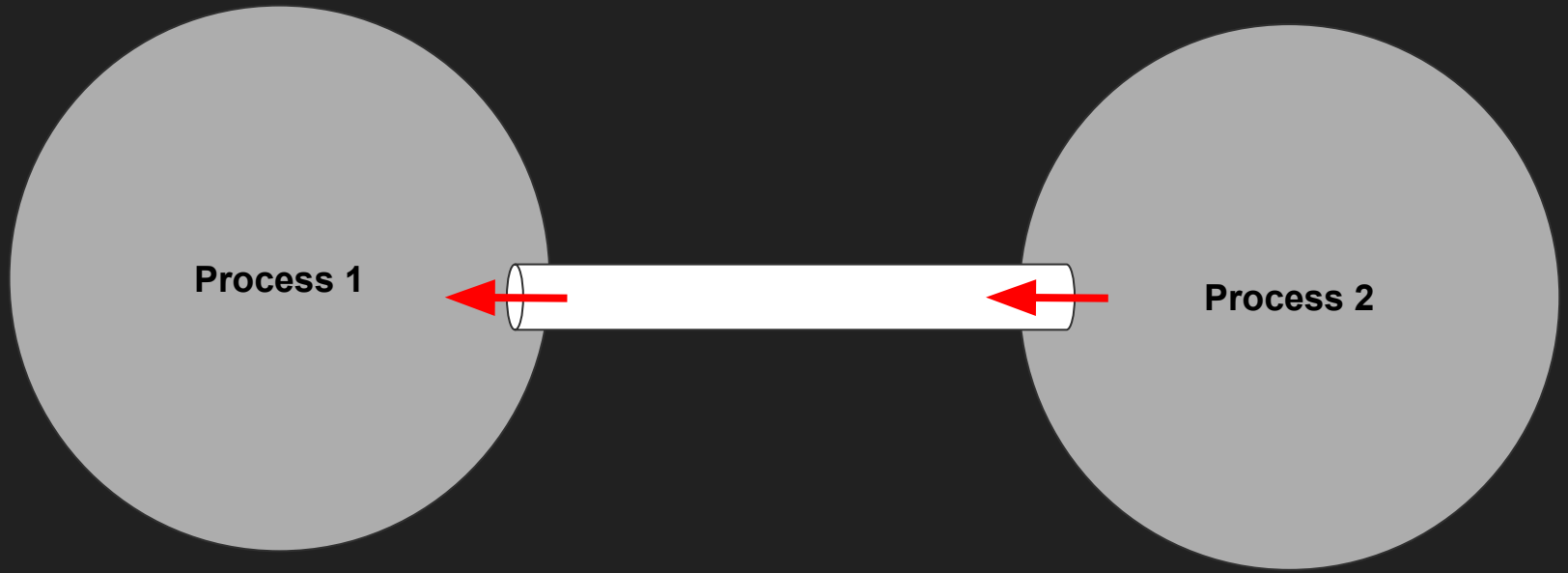
```
v = msgrcv( id,ptr,length,type,IPC_NOWAIT );
```

Focus of our lab: the IPC mechanism of Pipes

Pipes: Upstream/Downstream



Pipes: Upstream/Downstream



How we initialize pipes in C:

int fds[2];

pipes(fds);

Or

---

pipes(int fds[2])

Specifying upstream/downstream

fds[0] is the file descriptor on the downstream end of the pipe

- You can think of downstream as sending information down to a child process

fds[1] is the file descriptor on the upstream end of the pipe

- You can think of upstream as sending information up to the parent process

How we use pipes in C:

dup2(int old file descriptor, int new file descriptor)

Downstream:

dup2(fds[0], 0); //takes what it is going downstream and sends it to stdin

close(fds[1]); //closes the upstream end

Upstream:

dup2(fds[1], 1); //takes what is going upstream and sends it to stdout

close(fds[0]); //closes the downstream end

Additional Notes

1 = the file descriptor of stdout

0 = the file descriptor of stdin

Helpful commands to know

who: who the user is

sort: sorts symbols fist, then alphabetically but capital letters first

ls: lists all files/folders in the current folder you’re in

cat: “concatenate”

grep: used to search text

Checking Step 4

For Step 4:

Run the command in your terminal: `cat /etc/passwd | grep root`

See what the output is and compare it to your solution for step 4 to check if you correctly implemented your step 4 using pipes

For Step 5

Producer-Consumer Problem

As an idea: you can ask the user to input a sentence, have the producer send it to the consumer, and have the consumer then print it to see that it successfully sent

Implement using pipes

Your Lab Deliverables

Complete Steps 1-6 in the lab document, step 7 is a bonus

Once you're ready to demo all the steps let us know

What to Submit:

Code: For all the steps

Text File: Step 6 is the only step with a request to write down your observations and a question about what is happening in the code you run. If you do Step 7, the Bonus, then you'll also have to write a description of what your fix to Step 6 was.