# COMP7500/7506-Lecture 03: Shared-Memory Systems

**🟊: >85%, 🟊🟊: 70-85%, 🟊🟊🟊: 55-70%, 🟊🟊🟊🟊: 40-55%, 🟊🟊🟊🟊🟊: < 40%**

**🟊 Exercise 1:** (1.1) Which IPC model is shared memory and which one is message passing? (1.2) Which IPC model do you prefer? why? (1 Minute)

3_12.pdf

**🟊🟊🟊 Exercise 2 (Review):** Suppose two processes are exchanging a small amount of data, which interprocess communication model will you choose? Why?

1. Shared Memory
2. Shared Buffer
3. Message Passing
4. Message Chatting

**🟊🟊 Exercise 3 (Review):** Which interprocess communication model is faster than its counterpart? **Assumption:** message-passing systems are implemented using system calls; shared-memory regions are implemented in the user space.

1. Message Passing is faster
2. Shared Memory is faster
3. Shared Message is faster
4. Two models are both fast

**A picture containing indoor, table, wall

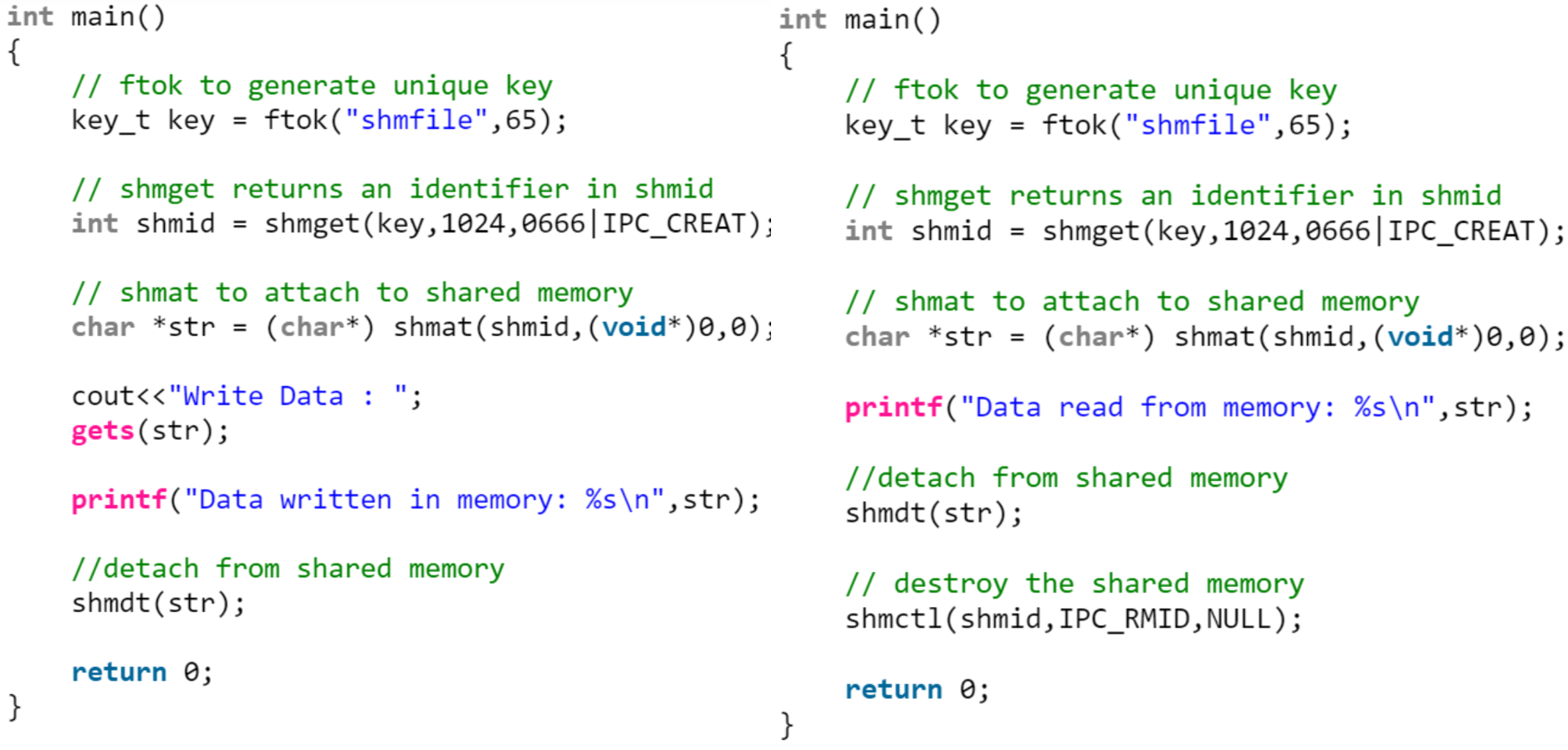
Description generated with very high confidence🟊🟊🟊 Exercise 4 (Review):** Which interprocess communication model is easier to implement in a distributed system?

1. Shared Memory is easier
2. Message Passing is faster
3. Message Passing is easier
4. Two models are equally easy

**🟊🟊 Exercise 5 (Plickers):** Where should the shared memory region reside? (30 Seconds)

1. Process A
2. Process B
3. Randomly pick process A or B
4. The process that creates the shared memory region

**🟊🟊🟊🟊 Exercise 6:** Please read the following sample source code. Can you design a diagram to intuitively illustrate the shared memory mechanism?



**🟊🟊🟊 Exercise 7:** Please complete the following code for the producer process.

item next\_produced;

while (true) {

/\* produce an item in next produced \*/

while (((in + 1) % BUFFER\_SIZE) == **\_\_\_\_\_\_\_\_\_\_**) /\*buffer is full\*/

; /\* do nothing \*/

buffer[in] = **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**;

in = **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**; /\*Hint: go to the next buffer\*/

}

**🟊🟊 Exercise 8:** Please complete the following code for the consumer process.

item next\_consumed;

while (true) {  
 while (in == **\_\_\_\_\_\_\_\_\_\_**)

; /\* do nothing \*/  
 next\_consumed = **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**;

out = **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**;

/\* consume the item in next consumed \*/

}

**🟊🟊🟊 Exercise 9 (Plickers):** What issue hasn’t been addressed in the above producer-consumer solution? (30 Seconds)

1. No issue. It is a perfect solution.
2. There a debugging problem
3. There is a synchronization problem
4. There is a user-space problem