ICS332: Inter-Process Communications (IPCs)

1. **Client-Server Communication - Java RMI: What is Java RMI?**
   Answer: "RPC in Java" in an object-oriented way. (Source: W4L2P31)

2. **Client-Server Communication - RMC: How does RMC work its "magic"?**
   Answer: Uses a client stub. (Source: W4L2P28)

3. **Client-Server Communication - RMC: Weak semantic: _____ (1), Stronger semantic: _____ (2).**
   Answer: (1) At most once. (2) Exactly once. (Source: W4L2P29)

4. **Client-Server Communication - Socket: 1) What is a socket? 2) What is a socket typically used for?**
   Answer: 1) A data communication endpoint so that two processes can communicate. 2) Typically used to communicate between two different hosts, but can work within a host (Source: W4L2P27)

5. **Client-Server Communication - UNIX Pipes: T/F - In Unix, a pipe is bi-directional.**
   Answer: False - In Unix, a pipe is MONO-DIRECTIONAL (write-end, read-end of a pipe) (Source: W4L2P33)

6. **Client-Server Communication - UNIX Pipes: What does the symbol | refer to in Linux?**
   Answer: The "pipe" command-line feature. (Source: W4L2P33)

7. **IPC - Blocking/Non-Blocking: State whether blocking & non-blocking are either synchronous or asynchronous.**
   Answer: > Blocking = synchronous (OS). > Non-blocking = asynchronous (OS). (Source: W4L2P19)

8. **IPC - Blocking/Non-Blocking: What does blocking send and blocking receive do?**
   Answer: > Blocking Send - sender blocks until the message is received. > Blocking Receive - receiver blocks until a message is available. (Source: W4L2P19)

9. **IPC - Blocking/Non-Blocking: What does non-blocking send and non-blocking receive do?**
   Answer: > Non-blocking Send - sender sends the message and continues. > Non-blocking receive - receiver receives a valid message or null. (Source: W4L2P19)

10. **IPC - Blocking/Non-Blocking: What is a rendez-vous communication style?**
    Answer: When both send and receive are blocking in communication. (Source: W4L2P19)

11. **IPC - Buffering: What is buffering (<- Answer 2)?**
    Answer: When messages are in transit, and still reside in "the link" of communication. (Source: W4L2P20)

    ** Question 2: What is being described above?

12. **IPC - Client-Server Communication: Provide (3) examples of this.**
    Answer: > Sockets > RPCs (Remote Procedure Calls) > Java RMI (Source: W4L2P26)

13. **IPC - Direct Communication: How does direct communication work (2)?**
    Answer: 1. Communication links between cooperating processes are established automatically and each link is associated with exactly one pair of communicating processes. 2. Processes communication with send() receive(). (Source: W4L2P15)

14. **IPC - Direct Communication: What is direct communication (<- Answer 2) (IPC)?**
    Answer: A form of communication between processes where each process must name the process they want to communicate with explicitly. (Source: W4L2P15)

    ** Question 2: What is being described above?

15. **IPC - Indirect Communication: How does indirect communication work (4)?**

Answer:
1. A process creates a mailbox.
2. The process shares the mailbox with the process it wants to communicate with.
3. Processes communicate through send and receive messages through mailbox.
4. The process who created the mailbox destroys the mailbox when communication end.
(Source: W4L2P16)

16. **IPC - Indirect Communication: How do you fix the mailbox sharing issue (3 solutions)?**

Answer;
> Allow a mailbox to be associates with at most two processes.
> Allow only one process at a time to execute a receive operation.
> Allow the system to select arbitrarily to the receiver.
(Source: W4L2P17)

17. **IPC - Indirect Communication: What is indirect communication (<- Answer 2)?**

Answer:
A form of communication between processes where each process communicate through mailboxes (i.e. ports). Processes can only communicate if they share a mailbox.
(Source: W4L2P16)

** Question 2:
What is being described above?

18. **IPC - Indirect Communication: What is the issue with mailbox sharing?**

Answer:
If a process shares a mailbox with multiple processes and then tries to send a message, there is no telling which process receives the message.
(Source: W4L2P17)

19. **IPC - Memory Copies: How do you reduce the number of memory copies (from user to kernel space)?**

Answer:
The kernel provides a send/recv abstraction that does take only pointers, DOES NOT DO ANY COPY (assume a shared-memory region is available).
(Source: W4L2P21)

20. **IPC - Message Passing: How do processes communicate to each other using message passing (2)?**

Answer:
1. Processes establish a communication "link".
2. Processes communicate suing send() and recv().

21. **IPC - Message Passing: Message passing is key for ___ (1) computing.**

Answer:
(1) Distributed
(Source: W4L2P9)

22. **IPC - Message Passing: T/F - Message passing also breaches the memory isolation principle.**

Answer:
False - Message passing does not have processes share any address space for communication, thus preserving the memory isolation principle.
(Source: W4L2P9)

23. **IPC - Message Passing: T/F - Message passing is typically blocking.**

Answer:
False - Message passing may be either blocking OR non-blocking.
(Source: W4L2P19)

24. **IPC - Message-Passing: What are (2) advantages and (2) disadvantages of message passing?**

Answer:
Advantage
---
1. Useful for exchanging small amounts of data.
2. Simple to implement in the OS.

Disadvantage
---
1. Cumbersome b/c code is sprinkled with send/recv OPs.
2. High-Overhead: one syscall per communication operation.
(Source: W4L2P4)

25. **IPC - Message Passing: What are (2) fundamental operations in message passing?**

Answer:
1. send
2. recv

26. **IPC - Message Passing: What is a big performance hit for message-passing?**

Answer:
Extra/Memory copies.
(Source: W4L2P24)

27. **IPC - Message Que: What are (3) typical message queue/capacity implementations?**

Answer:
1. Zero-capacity.
2. Bounded capacity.
3. Unbound capacity.
(Source: W4L2P20)

28. **IPC - Shared Memory:** **How do processes communicate to each other using shared memory?**	Answer: Processes communicate by reading/writing to the shared memory region. (Source: W4L2P5)
29. **IPC - Shared Memory:** **T/F - Shared Memory coding is very common in modern times.**	Answer: False. (Source: W4L2P8)
30. **IPC - Shared Memory:** **What are (2) advantages and (1) disadvantage of shared memory?**	Answer: Advantage ---------- 1. Low-Overhead: a few syscalls initially, then none. 2. More convenient for user (b/c read/write to/from RAM - like normal). Disadvantage ---------- 1. Harder to implement in the OS. (Source: W4L2P4)
31. **IPC - Shared Memory:** **What is contrary about shared memory to the memory protection idea central to multi-programming?**	Answer: The aspect of sharing memory creates vulnerabilities and lower efficiency (breaks memory isolation). (Source: W4L2P5, 8)
32. **IPC:** **What are (2) broad models of IPC?**	Answer: 1. Shared Memory. 2. Message Passing. (Source: W4L2P2)
33. **IPC:** **What are (4) reasons why cooperating processes exist?**	Answer: 1. Information sharing. 2. Computation speedup. 3. Modularity. 4. Convenience. (Source: W4L2P2)
34. **IPC:** **What is inter-process communication?**	Answer: Means of communication for cooperating processes. (Source: W4L2P2)