Interprocess Communication System (IPC system)

Implement an IPC system that lets processes communicate with one another by using *inter-process messages* according to the following scheme:

- A process wishing to send a message indicates the name of the process to which the message should be delivered, and text of the message. The IPC controller delivers this message to the destination process immediately if that process is waiting for a message; otherwise, the message remains *pending*.
- A process wishing to receive a message **does not** indicate the name of the process from which it wishes to receive a message. The IPC controller should block the process if no messages sent to it are still pending. Otherwise, the IPC controller delivers it the earliest message that was sent to it which is still pending, i.e., the IPC controller gives it messages sent by other processes in an FCFS manner.

You are required to code the IPC controller and *four* user processes as per the following specification:

1. Messages are of standard length only.
2. IPC controller has limited amount of buffer space available, say, enough buffer space to accommodate 20 messages. The IPC controller must make the best possible use of this space.
3. Messages will be handed over to a receiver process in an FCFS manner.
4. Operation of each user process is governed by a separate commands file. The user process will read the next command in the file, which will be either a *send* or a *receive* command with relevant details, and act on it. You may assume your own format for the commands.
5. After performing a send or receive request made by a process, the IPC controller will print a report line -

   Message sent/received: <Sender id>, <receiver id>, <message>

6. **IPC controller will detect deadlocks if they arise**. (A deadlock situation for a group of processes is a situation in which each of the processes in the group is blocked for an event that cannot occur. This way, processes that are in deadlock will remain blocked indefinitely.)

Points/Suggestions for implementation

1. Interpret the word 'process' in the above specification as a 'thread'.
2. Send/Receive actions performed by a user process are dictated by a commands file.
3. Each *send* or *receive* command will amount to a call to the IPC, made by the user process. Identity of the process making the call will be required by the IPC. You must make appropriate provision for it.
4. The sequence of *send* and *receive* commands for each process should be so chosen as to demonstrate working of the system in different situations. During viva, you will be required to demonstrate for a given situation.
5. You should shut down the IPC controller and the processes after some fixed number of messages have been exchanged.
6. **Your implementation must satisfy the following conditions:**
   (a) It should correctly implement all features of the specification.
   (b) It should have maximum concurrency.
   (c) It should not have busy waits.
   (d) It must detect all possible deadlock conditions.
7. It would be useful to
   (a) Analyze the problem specification and understand working of the IPC controller.
   (b) Analyze the synchronization requirements.
   (c) Decide how the synchornization should be prformed.
   (d) Code your solution and test it.