Summary:

As the threads (philosophers) go into the philosopher function they go into the take\_fork function. In this function is where the “state” of the philosopher becomes “HUNGRY”. Here the philosopher after being assigned to state HUNGRY goes through the test() function to see if he may pick up both forks on the left and right side and eat.

This test function checks the “left” and “right” side of the philosopher at the table and makes sure that neither of these philosophers are eating. If they are then his “request” to eat is ignored.

Here is where I got confused about what to do at first before I fully understood the problem. In the bounded buffer and reader and writer you would simply place the thread into a queue to wait until their turn. In the Dining Philosophers problem, it doesn’t work like that. Here the resources are set to the left and right of them and they can’t just be put into a queue. For instance, if philosopher 1 wants to eat BEFORE philosopher 4, it doesn’t make any difference since they are on total opposite sides of the table and the chopsticks (resources) they would use to eat with are totally different chopsticks (resources) so a queue makes no since.

This problem is actually resolved by the following: after a philosopher has passed the if condition in the test() statement and started eating and is done eating, he goes through the put\_fork() function where he actually puts both forks down. After this a the philosopher to the left of him is sent to the test() condition to make sure he is hungry, and the resources are available to be able to pick up both forks (chopsticks) to start eating. Then the same thing happens to the philosopher to the right of the current philosopher that is in focus.

This solution has nothing to do with a queue with any priority.