Mega-Prize Problem (Using shared memory synchronization with busy-waiting)

1. **Introduction:**
   Three persons $A$, $B$, and $C$ attend a game of housey with associated Mega-prizes. Three prizes are available, with the winner selecting one by choice, the second placed player selecting one of the remaining and the last player gets what is left. In the event of a tie, the prizes are grabbed in some arbitrary order. There is a judge $J$, whose role it is to call out the numbers one-by-one.

2. **Coupons:**
   The coupon associated with each player can be assumed to be a set of 7 numbers in the range $\{1, \ldots, 100\}$.

3. **Game:**
   The three players $A, B, C$ are ready with their respective coupons, and the housey game begins with the judge $J$ calling out numbers one-by-one.

4. **Rounds:**
   The game strictly follows a *round protocol*. After a number is called out, the three persons update their coupons by marking the number just called out. Only after the three persons have finished updating their coupons, does the judge call out the next number.

5. **Victory:**
   If all the numbers in the coupon for a person get marked, the player is declared victor, claims first right over prize and leaves the game. The others continue.

6. **Prize:** The equal right of access to prizes are to be shared among players finishing in the same round. Order of prize claims is to be maintained.

Write code for the processes $A, B, C, J$. The coupons are to be modelled as files containing lists of numbers. The required synchronization must be performed using shared variables with atomic reads and writes. The solution must be devoid of unnecessary-delays, and starvation.