L6: Store Queue

It’s the year 1984.
Due to ever-growing queues in stores, the Committee of the Polish United Workers' Party has decided to introduce self-service checkout counters.
These counters are meant to serve clients who come to the store to do their shopping.
Your task is to write software using message queues that will run on these checkouts.

Stages:

1. **(4 points)**
   Create a message queue with a **message length** of `MSG_SIZE` and a **maximum number of messages** as `MAX_MSG_COUNT`.
   Close it immediately after creating.
   Create `CLIENT_COUNT + 1` child processes.
   The first `CLIENT_COUNT` will act as clients, and the last one as the self-checkout.

   First, create the checkout process, wait 200 ms, and then start the clients.

   The checkout process prints the message “Store closing.” and ends.
   The child processes print “I will never come here. . . ” and end.

   The parent process waits for all child processes to end, prints “End”, removes the queue, and exits.

2. **(8 points)**
   Client processes, upon start, open the queue in blocking mode.
   Each sets a different random seed generator and draws a number `n` between `MIN_ITEMS` and `MAX_ITEMS`.
   Then `n` times it performs a loop where it:

   - Randomly selects:
     - `u`, an element from the array `UNITS`,
     - `p`, an element from the array `PRODUCTS`,
     - `a`, a number from the range [1, `MAX_AMOUNT`],
     - `P`, a priority equal to 0 or 1.
   - Sends a message to the queue with priority `P` and content “`<u><p> <a>`”, for example “7l vinegar”
   - If the message is not delivered to the queue within 1 second,
     then the process prints the message from the previous stage and breaks the loop.

   After the loop, the process closes the queue.

3. **(8 points)**
   The checkout process opens the queue in non-blocking mode.
   With 25% probability, it prints “Closed today”, does its cleanup, and exits.
   If that does not happen, it prints “Open today” and loops (`OPEN_FOR` times):

   - Prints “Hour `<h>`:00”, where `h` equals `START_TIME` + current iteration index (starting at zero).
   - Reads messages from the queue, sleeps 100 ms after reading each message.
   - Waits for 200 ms.

   If there was any message in the queue, the process checks its priority.
   If the priority is not zero, it prints “Please go to the end of the line!”.
   If the priority equals 0, it randomly prints one of the three messages:

   - “Come back tomorrow.”
   - “Out of stock.”
   - “There is an item in the packing zone that shouldn’t be there.”

   After completion, it performs cleanup (including closing the queue), prints “Store closing.”, waits 1s and exits (in this order!).

4. **(5 points)**
   Implement message reception in the checkout process via thread-based notifications.
   Leave only the time printing and waiting in the loop.

The first time you run your solution, you will likely encounter the error “Bad file descriptor” from `mq_receive` or `mq_notify`.
Consider where this error is coming from and handle it correctly—removing or moving `mq_close` in the checkout process is not allowed.