Project Goals

- Topic-Based Messaging : Implement the core publish-subscribe interaction pattern, focusing on the essential mechanisms of topic subscription, message publishing, and callback execution. We will implement a single, pre-defined topic ("goodbye_topic") for message exchange.
- Non-Blocking Message Broadcasting : Implement message delivery to the subscriber such that the publisher is not blocked. While full asynchronicity with separate threads isn't a primary goal *initially*, using the trap mechanism aims to prevent the publisher from waiting directly on the subscriber. The goal is to implement a message queue to send messages non-blocking.
- Efficient Message Handling: Prioritize the use of `container_alloc` and `container_free` for message and queue memory management to adhere to mCertikOS's resource quotas. Efficient message enqueue and dequeue are important but secondary to memory management constraints.
- Trap-Based Callback Execution : Leverage mCertikOS's trap handling mechanism to invoke subscriber callbacks. While OS-level threading *could* be used, the initial focus is on the trap mechanism for its integration with the OS and the project's constraints.
- Streamlined System Call Interface : Provide a straightforward set of system calls (`sys_pub`, `sys_sub`, `sys_unsub`) for publishers and subscribers to interact with the Pub/Sub system. The API should be easy to use and understand, reflecting the simplified nature of the implementation.

Implementation

Phase 1: Core Data Structures and Memory Management

- **Define Topic Registry**: Implement a hash map for topics, storing subscriber lists and queues.
- **Implement Message Queues**: Use circular buffers or linked lists, managed via `container_alloc` for memory allocation.
- **Integrate Container System**: Ensure each process's message queue adheres to its resource quota`1`.

Phase 2: Topic and Subscriber Management

- **Implement `topic_create`**: Add a new topic to the registry.
- **Implement `topic_subscribe`**:
  - Validate topic existence.
  - Allocate a message queue for the subscriber.
  - Store the subscriber’s callback and queue size.
- **Implement `topic_unsubscribe`**:
  - Remove the subscriber from the topic’s list.
  - Free the associated message queue and resources.

Phase 3: Syscall Integration

- **Add Syscall Definitions**: Modify `syscall.h` to include `sys_pub`, `sys_sub`, and `sys_unsub`.
- **Implement Syscall Handlers**:
  - **`sys_pub`**: Retrieve the message, iterate over subscribers, and enqueue the message.
  - **`sys_sub`/`sys_unsub`**: Manage subscriber registrations.
- **Argument Validation**: Ensure topics exist before allowing operations.

Phase 4: Message Delivery and Callbacks

- **Implement Message Broadcasting**:
  - For each subscriber, enqueue the message. If the queue is full, discard the oldest message.
- **Trigger Callback Execution**: Use mCertikOS’s trap handling (Lab 31) to schedule the callback in user space.
  - When a message is enqueued, send an interrupt or trigger a trap to the subscriber.
  - Execute the callback in a non-blocking thread or via asynchronous traps.

Prominent Functions

Core Functionalities of the System

- sys_sub: Subscribes to a topic, specifying a callback and queue size.
- sys_pub: Publishes a message to a topic.
- sys_unsub: Unsubscribes from a topic.

Implementation of Subscribers and Publishers

- topic_subscribe: Adds a subscriber to a topic's list, initializing their message queue.
- topic_unsubscribe: Removes a subscriber and frees associated resources.