# POSIX IPC (Inter-Process Communication) APIs

### Common Patterns Across IPC Mechanisms

- All three require file descriptor-based identifiers (shm\_open, sem\_open, mq\_open).  
- All use `open`-like functions to create or access resources.  
- All involve cleanup via unlink functions (shm\_unlink, sem\_unlink, mq\_unlink).  
- All rely on system-wide naming (`/name`) for process access.  
- All have file descriptor-based permissions and `O\_CREAT` flags.

## 1. Shared Memory (shm) - Fastest IPC for Data Sharing

### Steps:

1. Create/Open: shm\_open()  
2. Resize (if creating): ftruncate()  
3. Map to process memory: mmap()  
4. Use memory region (Read/Write)  
5. Detach from memory: munmap()  
6. Close descriptor: close()  
7. Remove shared memory: shm\_unlink()

### Key APIs:

- `int shm\_open(const char \*name, int oflag, mode\_t mode);`  
- `int ftruncate(int fd, off\_t length);`  
- `void \*mmap(void \*addr, size\_t length, int prot, int flags, int fd, off\_t offset);`  
- `int munmap(void \*addr, size\_t length);`  
- `int close(int fd);`  
- `int shm\_unlink(const char \*name);`

\*\*Use Case:\*\* Large data sharing, low overhead.

## 2. Semaphore (sem) - Synchronization

### Steps:

1. Create/Open: sem\_open()  
2. Initialize (if creating): sem\_init() (only for unnamed semaphores)  
3. Wait (lock/decrement): sem\_wait()  
4. Signal (release/increment): sem\_post()  
5. Close descriptor: sem\_close()  
6. Remove semaphore: sem\_unlink()

### Key APIs:

- `sem\_t \*sem\_open(const char \*name, int oflag, mode\_t mode, unsigned int value);`  
- `int sem\_wait(sem\_t \*sem);`  
- `int sem\_trywait(sem\_t \*sem);`  
- `int sem\_post(sem\_t \*sem);`  
- `int sem\_close(sem\_t \*sem);`  
- `int sem\_unlink(const char \*name);`

\*\*Use Case:\*\* Process synchronization, avoiding race conditions.

## 3. Message Queue (mq) - Structured Message Passing

### Steps:

1. Create/Open: mq\_open()  
2. Send a message: mq\_send()  
3. Receive a message: mq\_receive()  
4. Get queue attributes: mq\_getattr()  
5. Close queue descriptor: mq\_close()  
6. Remove message queue: mq\_unlink()

### Key APIs:

- `mqd\_t mq\_open(const char \*name, int oflag, mode\_t mode, struct mq\_attr \*attr);`  
- `int mq\_send(mqd\_t mqdes, const char \*msg\_ptr, size\_t msg\_len, unsigned int msg\_prio);`  
- `ssize\_t mq\_receive(mqd\_t mqdes, char \*msg\_ptr, size\_t msg\_len, unsigned int \*msg\_prio);`  
- `int mq\_close(mqd\_t mqdes);`  
- `int mq\_unlink(const char \*name);`

\*\*Use Case:\*\* Ordered, structured message-based IPC.

## Comparison Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IPC Type | Creation API | Read API | Write API | Close API | Remove API |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| Shared Memory | shm\_open() | Direct Memory Access | Direct Memory Access | munmap(), close() | shm\_unlink() |
| Semaphore | sem\_open() | sem\_wait() | sem\_post() | sem\_close() | sem\_unlink() |
| Message Queue | mq\_open() | mq\_receive() | mq\_send() | mq\_close() | mq\_unlink() |

## Summary of Similarities

- All require open(), close(), and unlink()  
- All involve file descriptor-based IPC  
- All need explicit cleanup (unmap, unlink, close)  
- All have named and unnamed versions  
- All work across processes in Linux