Message Queues

UNX511 Week 10 Class 1

John Sellens

July 15, 2025

Seneca Polytechnic

Outline

Message Queues

Message Queues

Message Queues

- Another inter-process communication (IPC) method
- Originated in UNIX System V
- Between cooperating not-necessarily-related processes
- Queue senders and receivers – can be multiples
- Like files, a queue has an owner and permissions
  - But doesn't appear in the file system
- Other System V IPC: shared memory, and semaphores
  - Discussed later in the course
- Some similarities with pipes and FIFOs

msgget() – Create a Message Queue

- Takes a key and flags, returns message queue identifier
  - Similar in function to a file descriptor
- Key is an encoded “pathname” (which seems to not need to exist)
- Flags include permission bits and queue options
- A message queue has a fixed (but modifiable) capacity
- See `msgget(2)` and `ftok(3)` and the code samples
- Will continue to exist until explicitly removed (like FIFOs)

Message Format

- A message is a caller-defined struct (chunk of memory)
- The first long int's worth of space is treated as the message type
  - Message type can be an ID – for processes, workers, types of messages, ...
  - Or a priority level – higher numbers can be treated as lower priority
- The remainder of the memory is free-form
- Messages are sent and received in their entirety
  - No partial sends or receives
- See `msgop(2)` for `msgsnd()` and `msgrcv()`

msgsnd() – Send a Message

- Takes message queue ID, memory, size, and flags
- If queue would be overfull, `msgsnd()` blocks until space available
  - Unless the `IPC_NOWAIT` flag is used
- Remember that the first “long” is the message type

msgrcv() – Receive a Message

- Takes message queue ID, memory, size, message type, and flags
  - Takes a copy of an item in the queue, which is then removed
- If no message available, `msgrcv()` blocks until one is available
  - Unless the `IPC_NOWAIT` flag is used
- Message type determines which message is returned
  - First message in queue
  - First message with (or without) specific type
  - First message with type less than or equal to a threshold
  - This means: not a very strict queue
  - See the documentation: `msgop(2)`
- Fails if you didn't pass a big enough buffer to hold the message

msgctl() – Control a Queue

- Get or set information about a queue
- Can delete a queue
- See `msgctl(2)`
- See also the command `ipcs(1)`

Queue Fullness

- If a queue gets full, processing can be blocked
- Our examples use threads to try to read from the queue as soon as possible
  - And keep the queue in the kernel as empty as possible
- That might be overkill in some situations

Message Queue Code Examples

- Let's have a look in unx511_samples
  - `https://github.com/jsellens/unx511_samples`
- `week10_1/1_basic` – basic message queues
- `week10_1/2_nonblocking` – non-blocking message queues
- `week10_1/3_messagetypes` – different message types
- `week10_1/4_complex` – more complex data loads
- `week10_1/5_msgctl` – retrieve message queue information

Summary

- Part of the System V IPC family
- There are also POSIX message queues
- Can be useful for specific purposes
- But a UNIX domain or IP datagram IPC method is much more common
- There are indications that this is “not the fastest” IPC method