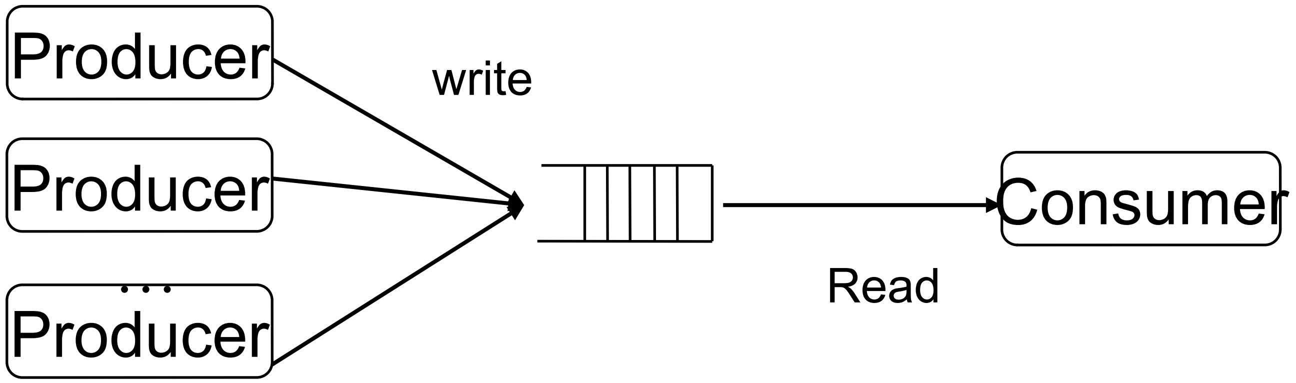
Message Queues

- SV

- POSIX

Message Queues

- Link between producer and consumer is indirect
  - Producer write messages on the queue
  - Without selecting consumer
- Consumer retrieve a message from the Queue
  - Without selecting producer
- Writings are unblocking
  - As long as resources are available
- Reading is blocking
  - When queue is empty



Message queues

- A Message Queue is a linked list of message structures
  - stored inside the kernel’s memory space and accessible by multiple processes
- Synchronization is provided automatically by the kernel
- Preallocated message buffers
- Messages with priority.
  - A message with a higher priority is always received first.
- Send and receive functions are synchronous by default.
  - Possibility to set a wait timeout to avoid nondeterminism.
- Support asynchronous delivery notifications.

Programming steps

- Define Message structure
- Create message Queue
- Connect to queue
  -
- Read messages
-
-
- Close Queue
- Destroy Queue

- Connect to queue
- Write Messages
- Close Queue

Message Structure

- Multiple processes should agree on the message structure/size
  - Application level
  - Message queues handle different sizes
- Programmer should define a structure
  - With pre-defined size
  - Able to accommodate multiple sub-types

SV MQ - Message Structure

- Example:
  - struct mymsg {
  - **long msg_type;**
  - char mytext[512]; /* rest of message */
  - int somethingelse;
  - };
- msg_type used in reception

SV MQ – creation

- msgget - get a System V message queue identifier
  - int msgget(key_t key, int msgflg);
- Create Private
  - Key – IPC_PRIVATE
  - Hinerited by chld processes
- Create Public
  - Key – not in use (ipcs)
  - Msgflag - IPC_CREAT
  - Verify if existes
    - Msgflag - O_CREAT | O_EXCL

SV MQ – opening

- msgget - get a System V message queue identifier
  - int msgget(key_t key, int msgflg);
- Open a Public MQ
  - Key – already creates MQ
  - Msgflag - NULL

SV MQ - write

- int msgsnd(int msqid,
- const void *msgp, size_t msgsz,
- int msgflg);
  - Writes a message to the queue
  - Parameters
    - Msqid – quue id (returned from msgget)
    - Message + size
    - Msgflags - IPC_NOWAIT

SV MQ - Read

- ssize_t msgrcv(int msqid,
- void *msgp, size_t msgsz,
- long msgtyp, int msgflg);
  - Reads a message from queue
  - Parameters
    - Msqid – queue id (returned from msgget)
    - Pointer to buufer + max size size
    - Type of message
    - Msgflags - IPC_NOWAIT

SV MQ - Read

- ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg);
  - MsgType
    - 0 – first message
    - > 0 – first message with that type
    - <0 - first message with
      - the lowest type less than or equal to the absolute value of msgtyp
  - Msgflag
    - IPC_NOWAIT
    - MSG_COPY – does nt remove message

SV MQ - destruction

- int msgctl(int msqid, int cmd, struct msqid_ds *buf);
  - Msqid
  - Cmd – IPC_RMID
  - msqid_ds NULL

POSIX MQ

POSIX MQ - Message Structure

- Array of bytes
- Priority / message selection
  - API
- Each message has an associated priority,
- Messages are always delivered to the receiving process highest priority first.
- Message priorities range
  - From 0 (low) to sysconf(_SC_MQ_PRIO_MAX) - 1 (high).
  - On Linux, sysconf(_SC_MQ_PRIO_MAX) returns 32768,
  - POSIX.1 requires a range from 0 to to 31

-

POSIX MQ – creation

- mq_open - open a message queue
  - mqd_t mq_open(const char *name,
  -                                 int oflag, mode_t mode,
  -                                 struct mq_attr *attr);
- Name – identifier
- Oflags -
  - O_CREAT | O_RDONLY | O_WRONLY | O_RDWR
- Mode
  - File access modes rwx / ugw 0666

POSIX MQ – creation

- mq_open - open a message queue
  - mqd_t mq_open(const char *name,
  - int oflag, mode_t mode,
  - struct mq_attr *attr);
- attr
  - NULL
  - struct mq_attr queue_attr;
    - queue_attr.mq_maxmsg = 16;
    - queue_attr.mq_msgsize = 128;

POSIX MQ – opening

- mq_open - open a message queue
  - mqd_t mq_open(const char *name, int oflag)
- Default settings
  - Name – identifier
  - Oflags -
    - O_RDONLY O_WRONLY O_RDWR

POSIX MQ - mq_open

- Creates
  - O_CREAT
- Message queue is assigned to a file
  - In /dev/msgque/
  - File name is used by other processes

- mq_close
  - close a message queue descriptor
  - Process can no longer use queue

- mq_unlink
  - removes a message queue
  - Deletes the file

POSIX MQ - write

- int mq_send(mqd_t mqdes,
- const char *msg_ptr, size_t msg_len,
- unsigned int msg_prio);
  - Writes a message to the queue
  - Parameters
    - mqdes – queue id (returned from mq_open)
    - Message + size
    - msg_priority – udes in mq_receive

POSIX MQ - read

- ssize_t mq_receive(mqd_t mqdes,
- char *msg_ptr, size_t msg_len,
- unsigned int *msg_prio);
- Reads a message from the queue
  - mqdes – queue id (returned from mq_open)
  - Message + buffer size
  - msg_priority – used in mq_receive

POSIX MQ - read

- ssize_t mq_receive(mqd_t mqdes, char *msg_ptr, size_t msg_len,unsigned int *msg_prio);
  - Messages are always delivered to the receiving process highest priority first.
  - msg_priority –
    - NULL
    - Not NULL stores the prioryte of received message

Read/write

- Empty Queue
  - Receive Call blocks
  - mq_timedreceive
    - Block some time
- Full queue
  - Send blocks
  - mq_timedsend
    - Blocks some time
- … , const struct timespec *abs_timeout);
- Errno - ETIMEDOUT

POSIX MQ - limits

- On the user program
  - queue_attr.mq_maxmsg = 16;
  - queue_attr.mq_msgsize = 128;
- Values limited by the OS
  - /proc/sys/fs/mqueue/
- Change on:
  - /etc/security/limits.conf

Message Queues

- Implementation – Kernel / syscall
- Scope - local
- No Duplex
- Time-uncoupling
- Space-uncoupling
- Explicit
- Synchronization – Yes (reads) no (writes)
- Process relation - unrelated
- Identification –SV – integer   POSIX - string
- API – specific API