message queues

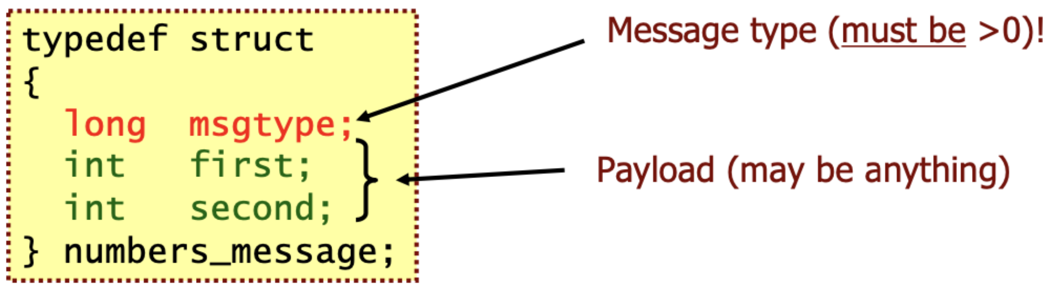
- **STREAMS** represent a flow of bytes. There are no fixed data boundaries.
  - The sender requests the transmission of N bytes
  - The data starts flowing, the receiver starts getting it
  - The receiver may get several chunks of less then N bytes

- **MESSAGES** represent a complete fixed structure of data
  - It is like sending a letter. Either you get if fully or you do not. You do not get half a letter.

- message queues are **asynchronous**
  - a process can start executing, write some messages to a message queue and die. later, another process can come alive and receive them
    - does not require that both sender and receiver are present at the same time
  - message queues are maintained by the operating system. they are not destroyed if a process dies

message payload

- messages can be anything but **must always have a *long* integer in the beginning - message type identifier**



```
typedef struct
{
  long  msgtype;
  int   first;
  int   second;
} numbers_message;
```

functions - System V IPCs

```c
int msgget(key_t key, int flags);
// obtains an identifier to an existing message queue or creates a new one
```

```
- key:
        - can be `IPC_PRIVATE`: creates a new unique identifier
        - can be an existing identifier
        - `ftok()` can be used to generate a number based on a filename
- flags:
        - `IPC_CREAT`: creates a new message queue
        - permission in octal (i.e. `IPC_CREAT | 0777`)
- returns de message queue identifier (or -1 on error)
```

```c
int msgctl(int msqid, int cmd, struct msqid_ds *buff);
// provides a variety of control operations on the message queue
```

```
- msqid: values returned by `msgget()`
- cmd: the command itself
        - `IPC_RMID`: remove the message queue
- buff: structure used in some control operations
```

```c
int msgsnd(int msqid, const void *message, size_t length, int flags);
// puts a message in a message queue
// appends a copy of the messsage to the message queue specified
```

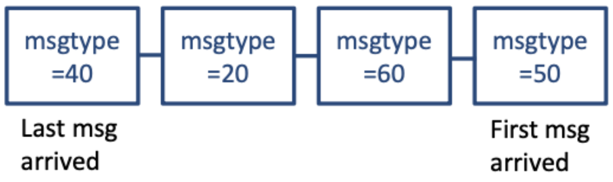
```
- msqid: values returned by `msgget()`
- length: the size of the payload (not the size of the entire message)
- flags: 0 or `IPC_NOWAIT`(non-blocking)
- on error, returns -1
```

⚠ **Warning**

the calling process must have write permissions on the message queue in order to send a message

```c
int msgrcv(int msgid, void *message, size_t length, long msgtype, int
flags);
// retrieves a message from the message queue with identifier `msgid` and
places it in the buffer pointed to by `message`
```

```
- length: maximum payload (bytes) we are willing to receive
- msgtype: type of message to receive
        - 0: the first message in the queue is returned (FIFO)
        - > 0: the first message of type `msgtype` is retrieved
        - < 0: the first message in the queue with the lowest type lesss
than or equal to the absolute value of `msgtype` will be read
- flags: 0 or IPC_NOWAIT (non-blocking)
- on error, returns -1
```



Note: This example supposes that each function is executed with the initial MSQ messages

- `msgrcv(id,&msg,size,20,0);`
  - Returns message with msgtype=20
- `msgrcv(id,&msg,size,10,0);`
  - Blocks waiting for message with msgtype=10
- `msgrcv(id,&msg,size,0,0);`
  - Returns first message: the one with msgtype=50
- `msgrcv(id,&msg,size,-50,0);`
  - Returns message with the lowest type <= |-50| ; returns message with msgtype=20

⚠ **Warning**

the calling process must have read permissions on the message queue in order to receive a message