read() will slightly wait for the next characters to be written to the pipe.

Process Management 2.11.2022
Wednesday

→ IPC
→ Process Synchronization

pipes and FIFOs → communication is in the form of byte streams.

No concept of boundaries among data.
→ Revitalize the concept / system calls related to files.

Data in many cases should be divided into blocks or messages.

A new way of IPC called message queues is used. Because files do not need to be

have support for message blocks / data blocks, new system calls / library functions are needed.

mqd_t mq_open (char *name, int oflags, ... mode, msg_attr);

Message queues have a specific data structure calls attributes, which govern their behavior. Some of the attributes can be changed after creation, but many of them become fixed.

```
struct msg_attr {
    long mq_flags;
    long mq_maxmsg;   /* No. of messages allowed */
    long mq_msgsize;  /* in bytes */
    long mq_curmsgs;  /* No. of current messages in the queue */
};
```

Size of queue

```
struct msg_attr my_msg_attr;
my_msg_attr.mq_maxmsg = 100;
my_msg_attr.mq_msgsize = 1024;
mqd_t mymqd = mq_open("abc",
                      O_RDWR;
                      NULL; &my_msg_attr
```

/* mq_setattr (*attr); Prototype */

mq_setattr (&my_msg_attr);

↳ can be used to change the message queue attributes

mq_getattr (mqd_t)

mq_send (mqd, *msg_ptr, msg_len, msg_prio);

Message queues are actually priority ques, where messages are received in the order of priority. If the same priority is given to all, then it keeps a first in first out data structure

ssize_t mq_receive (mqd, *msg_ptr, msg_len, *msg_prio)

↓

Returns the number of bytes in the message

gets pointed to a location storing the priority value of the received message.

msg_notify (mqd_t mqdes, *notification)

↳ Alternative technique to know when message is available in the queue.

```
mqd_t mydt = mq_open ("abc", ...)
while (mq_receive (mydt, &msg, 100..)) {
        /* Process the message */
}
```

Do a message receive only after getting the notification to avoid waiting for message.
This is called non-blocking read / receiving of messages.

mq_close (mqd_t, mqdes);
↳ If writing to message queue stops

mq_unlink (char *name);
↳ Stop using the message-queue.
Only when all processes using the message queue name called unlink, and whoever had created it has also closed it, the message queue data structure can be destroyed by the kernel.
Not all data can be sent because sending memory address has no meaning.

Sending or receiving message is slower than reading or writing to pipes, since messages need to maintain consistency.

Sockets

Another form of communication mechanism.

↳ Can send messages or bytestreams. (datagrams)

- ↓ No sequence among messages or datagrams is imposed by sockets
- ↓ very similar to sending data over pipes

These sockets can be used for both communication within a single system and across systems.

↳ need some way of initiating communication. There is a protocol needed for such type of communication - UNIX sockets, TCP/IP sockets

Clients initiate communication, whereas servers are ready to accept initiation requests