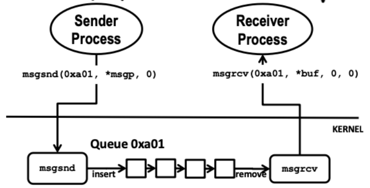
message queues

* like pipes, but in pipes you cannot know wheter sender has written all its message before receiver reads it.



- in message queues messages are put in one batch / block
- you cannot send the half of the message, no worry
- processes share common key in order to access the pipe

non-blocking message passing = asynchronous, when you try to receive a message, if there is none, return immediately

blocking message passing = synchronous, when you try to receive a message, if there is none, you are blocked by sender until the message is transferred

message sending

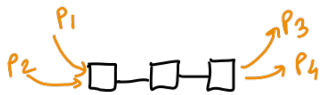
```c
#define MSGSZ     128
typedef struct msgbuf { /*msg structure */
        long    mtype;
        char    mtext[MSGSZ];
        } message_buf;
main(){
    int result, msgid, msgflg = IPC_CREAT | 0666;
    key_t key;
    message_buf sbuf;
    size_t buf_length;

    key = 1234;
    msqid = msgget(key, msgflg);
    sbuf.mtype = 1; /*send a msg of type 1 */
    strcpy(sbuf.mtext, "Did you get this?");
    buf_length = strlen(sbuf.mtext) + 1 ;
    /* send msg */
    result = msgsnd(msqid, &sbuf, buf_length, IPC_NOWAIT);
    printf("Message: \"%s\" Sent\n", sbuf.mtext);
    exit(0);
}
```

message receiving

```c
#define MSGSZ     128
typedef struct msgbuf { /* msg struct */
    long    mtype;
    char    mtext[MSGSZ];
} message_buf;
main(){
    int msqid;
    key_t key;
    message_buf  rbuf;
    key = 1234;
    msqid = msgget(key, 0666);
    /* receive msg */
    result = msgrcv(msqid, &rbuf, MSGSZ, 1, 0);
    /* print msg */
    printf("%s\n", rbuf.mtext);
    exit(0);
}
```

- cannot broadcast



- specific message type can be defined to send to send it to specific procces