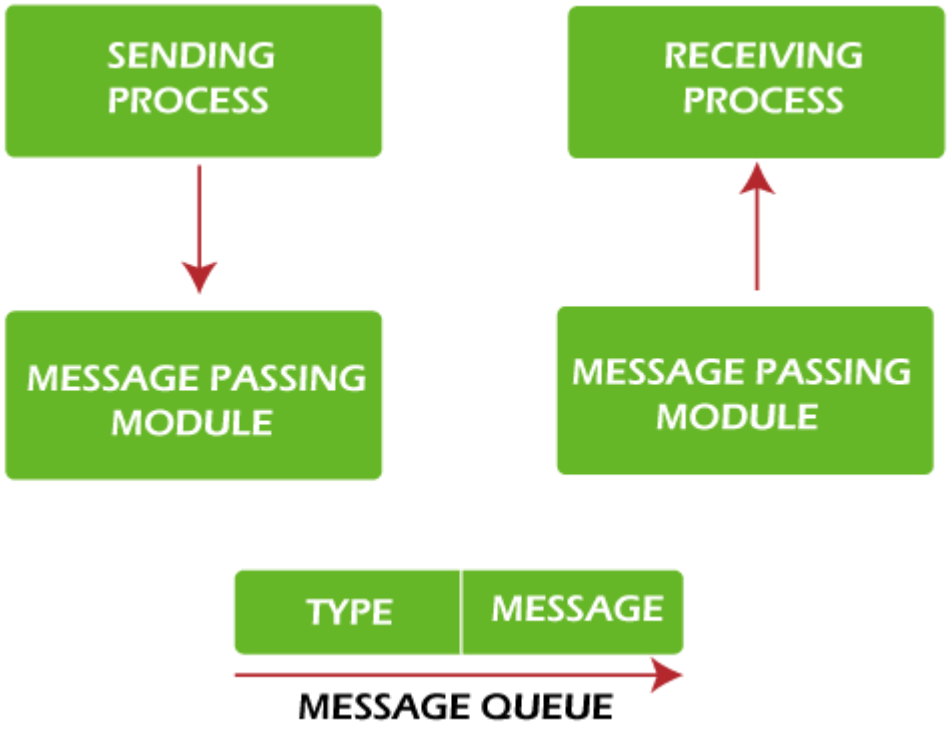
Operating Systems (CS F372)
Tutorial Sheet 4
IPC – Message Queues

In this tutorial sheet, we shall explore the concepts of message queues in Inter-Process Communication (IPC).

Message queues provide an additional technique for IPC. The main advantage of using Message Queues is that they provide "Asynchronous Communication Protocols" i.e., the sender and the receiver do not need to be active at the same time. Two (or more) processes can exchange information via access to a common system message queue. The *sending* process places via some (OS) message-passing module a message onto a queue which can be read by another process. Each message is given an identification or type so that processes can select the appropriate message. Processes must share a common key in order to gain access to the queue in the first place.



Basic Message Passing IPC messaging lets processes send and receive messages, and queue messages for processing in an arbitrary order. Unlike the file byte-stream data flow of pipes, each IPC message has an explicit length. Messages can be assigned a specific type. Because of this, a server process can direct message traffic between clients

on its queue by using the client process PID as the message type. For single-message transactions, multiple server processes can work in parallel on transactions sent to a shared message queue.
Before a process can send or receive a message, the queue must be initialized (through the msgget() function). Operations to send and receive messages are performed by the msgsnd() and msgrcv() functions, respectively.

Question 1:

1. Write a C program to create a message queue and send a message containing your name and BITS ID.

2. Write another C program to read messages from the message queue and print them to stdout.

Try it yourself: What happens when you try to read from an empty message queue?

Question 2:

1. Write a C program to create a message queue and send three messages containing "message1", "message2" and "message3" in that order.

2. Write another C program to read the messages in reverse order ("message3", "message2", "message1").
   Hint: Use msgtype while sending and receiving messages.

Try it yourself: What happens when you pass a negative integer as msgtype to msgrcv()?

Hint: read through the manual page for msgrcv