Session - 31

Message Queues – Inter-Process Communication (IPC) Mechanism

Message Queues are an IPC mechanism that allows **asynchronous communication** between processes by sending and receiving **structured messages** through a **queue maintained by the kernel**.

🔹 What is a Message Queue?

A **Message Queue**:

- Is a **kernel-managed list of messages**.
- Allows one process to **send messages** to the queue.
- Allows another process to **receive messages** from the queue.
- Is **persistent** (can exist after processes terminate unless explicitly removed).
- Provides **better control** compared to pipes (like message prioritization and message boundaries).

🔸 Key System Calls (System V Message Queues)

Function	Purpose
msgget()	Create a message queue or get ID
msgsnd()	Send a message to the queue
msgrcv()	Receive a message from the queue

Function Purpose

msgctl() Control operations (delete, get info)

🔧 Message Queue Example in C

📦 Define Message Structure

```
struct msgbuf {
    long mtype;       // Message type (must be > 0)
    char mtext[100];    // Message text
};
```

👨‍💻 Sender Code

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <string.h>

struct msgbuf {
    long mtype;
    char mtext[100];
};
```

```
int main() {
    key_t key = ftok("progfile", 65);          // Generate unique key
    int msgid = msgget(key, 0666 | IPC_CREAT);  // Create message queue

    struct msgbuf message;
    message.mtype = 1;
    strcpy(message.mtext, "Hello from sender");

    msgsnd(msgid, &message, sizeof(message.mtext), 0); // Send message

    printf("Message sent: %s\n", message.mtext);
    return 0;
}
```

👩‍💻 **Receiver Code**

```
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct msgbuf {
    long mtype;
    char mtext[100];
};
```

```
int main() {
    key_t key = ftok("progfile", 65);          // Generate same key
    int msgid = msgget(key, 0666 | IPC_CREAT);  // Access existing queue

    struct msgbuf message;
    msgrcv(msgid, &message, sizeof(message.mtext), 1, 0); // Receive message

    printf("Message received: %s\n", message.mtext);

    msgctl(msgid, IPC_RMID, NULL);  // Delete message queue
    return 0;
}
```

📦 Features

- **Message boundaries preserved** (unlike pipes).
- Can send and receive **messages of different types**.
- Messages are **asynchronous**—no need for sender and receiver to be active at the same time.
- Supports **priority/message type filtering**.

📊 Advantages

- More control (types, order, etc.)
- Better for structured communication.
- Can be used by **multiple unrelated processes**.

🚫 Limitations

- Limited queue size and message size (system dependent).
- More overhead than pipes.
- Slightly more complex API.
- Not as portable across systems as POSIX message queues.

🧹 Clean-Up

Always remove message queue after use:

msgctl(msgid, IPC_RMID, NULL);