Session - 32

Shared Memory – Inter-Process Communication (IPC) Mechanism

Shared memory is the **fastest** IPC mechanism. It allows **two or more processes** to communicate by **accessing a common memory space** directly.

🔹 What is Shared Memory?

- A memory segment is created by one process.
- Other processes **attach to this segment** to read/write.
- **No kernel involvement during data access**, making it faster than pipes or message queues.
- Often used with **semaphores** for synchronization (to avoid race conditions).

🔧 System V Shared Memory – Key Functions

Function	Purpose
shmget()	Create/get shared memory segment
shmat()	Attach shared memory to process address space
shmdt()	Detach shared memory
shmctl()	Control operations (delete, get info)

🧪 **Example: Shared Memory in C**

📜 **Writer Process (Creates and writes to memory)**

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

int main() {
    key_t key = ftok("shmfile", 65);            // Generate unique key
    int shmid = shmget(key, 1024, 0666 | IPC_CREAT);  // Create shared memory

    char *str = (char*) shmat(shmid, NULL, 0);   // Attach to memory
    strcpy(str, "Hello from writer");           // Write to shared memory

    printf("Data written in memory: %s\n", str);
    shmdt(str);  // Detach from shared memory
    return 0;
}
```

📖 Reader Process (Reads from shared memory)

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

int main() {
    key_t key = ftok("shmfile", 65);            // Use same key
    int shmid = shmget(key, 1024, 0666);        // Get shared memory ID

    char *str = (char*) shmat(shmid, NULL, 0);   // Attach to memory
    printf("Data read from memory: %s\n", str);

    shmdt(str);  // Detach from shared memory
    shmctl(shmid, IPC_RMID, NULL);  // Destroy shared memory
    return 0;
}
```

📦 Features of Shared Memory

- 🔁 **Bi-directional communication** (read/write)
- ⚡ **Extremely fast** since data doesn't pass through the kernel
- 📍 **Requires synchronization** (like semaphores/mutex) for safe concurrent access

🆚 Shared Memory vs Other IPC Mechanisms

Feature	Shared Memory	Pipes	Message Queues
Speed	⭐ Very fast	Moderate	Slower
Communication	Bi-directional	Unidirectional	Asynchronous
Kernel Involved	Only setup	Yes (data transfer)	Yes (data handling)
Synchronization	Manual (needed)	Built-in	Built-in
Message Boundaries	No	No	Yes

🚫 Limitations

- Complex to use correctly (needs explicit **synchronization**).
- Risk of **race conditions** and **data corruption** without locks.
- Memory is **not automatically cleaned** — must explicitly remove with shmctl.