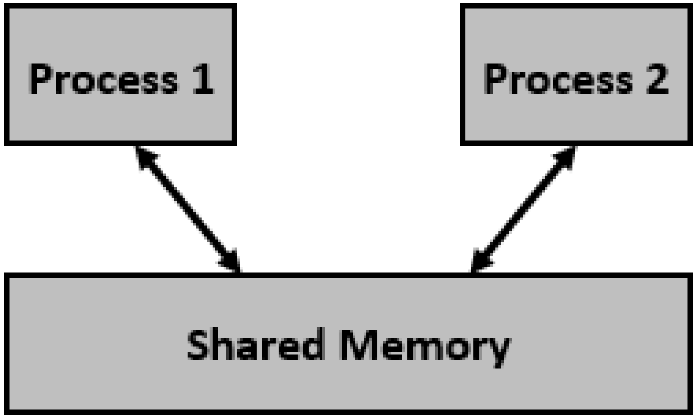
Shared Memory

What is shared memory?

- Shared memory is an IPC mechanism
- Shared memory is an OS feature that allows multiple processes to access and modify a shared memory region simultaneously
- This allows processes to communicate with each other and share data efficiently
- Here, two or more process can access the common memory and communication is done via this shared memory where changes made by one process can be viewed by another process



Issues with pipes, message queues

For two process to exchange information, the information has to go through the kernel

- An example scenario:
  - ❑ Server reads from the input file.
  - ❑ The server writes this data in a message using either a pipe, fifo or message queue.
  - ❑ The client reads the data from the IPC channel, again requiring the data to be copied from kernel's IPC buffer to the client's buffer
  - ❑ Finally, the data is copied from the client's buffer
- A total of four copies of data are required (2 read and 2 write)
- Shared memory provides a way by letting two or more processes share a memory segment
- With Shared Memory the data is only copied twice – from input file into shared memory and from shared memory to the output file

Sys calls used in shared memory implementation

Function	Signature	Description
ftok()	key_t ftok()	It is used to generate a unique key.
shmget()	int shmget(key_t key,size_t size, int shmflg);	Upon successful completion, shmget() returns an identifier for the shared memory segment.
shmat()	void *shmat(int shmid ,void *shmaddr ,int shmflg);	Before you can use a shared memory segment, you have to attach yourself to it using shmat(). Here, shmid is a shared memory ID and shmaddr specifies the specific address to use but we should set it to zero and OS will automatically choose the address.
shmdt()	int shmdt(void *shmaddr);	When you are done with the shared memory segment, your program should detach itself from it using shmdt().
shmctl()	shmctl(int shmid,IPC_RMID,NULL);	When you detach from shared memory, it is not destroyed. So, to destroy shmctl() is used.

Steps in using shared memory as an IPC mechanism

1. Create the shared memory segment or use an already created shared memory segment (shmget())
2. Attach the process to the already created shared memory segment (shmat())
3. Detach the process from the already attached shared memory segment (shmdt())
4. Control operations on the shared memory segment (shmctl())

Shared memory for writer process

```
#include <iostream>
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
using namespace std;

int main()
{
  // ftok to generate unique key
  key_t key = ftok("shmfile", 65);

  // shmget returns an identifier in shmid
  int shmid = shmget(key, 1024, 0666 | IPC_CREAT);

  // shmat to attach to shared memory
  char* str = (char*)shmat(shmid, (void*)0, 0);

  cout << "Write Data : ";
  cin.getline(str, 1024);

  cout << "Data written in memory: " << str << endl;

  // detach from shared memory
  shmdt(str);

  return 0;
}
```

Shared memory for reader process

```
#include <iostream>
#include <stdio.h>
#include <sys/ipc.h>
#include <sys/shm.h>
using namespace std;

int main()
{
  // ftok to generate unique key
  key_t key = ftok("shmfile", 65);

  // shmget returns an identifier in shmid
  int shmid = shmget(key, 1024, 0666 | IPC_CREAT);

  // shmat to attach to shared memory
  char* str = (char*)shmat(shmid, (void*)0, 0);

  cout << "Data read from memory:" << str;

  // detach from shared memory
  shmdt(str);

  // destroy the shared memory
  shmctl(shmid, IPC_RMID, NULL);

  return 0;
}
```

`int shmget(key_t key, size_t size, int shmflg)`

- ❑ The **first argument, key,** recognizes the shared memory segment. The key can be either an arbitrary value or one that can be derived from the library function ftok(). The key can also be IPC_PRIVATE, means, running processes as server and client (parent and child relationship) i.e., inter-related process communication. If the client wants to use shared memory with this key, then it must be a child process of the server. Also, the child process needs to be created after the parent has obtained a shared memory.

- ❑ The **second argument, size,** is the size of the shared memory segment rounded to multiple of PAGE_SIZE

- ❑ The **third argument, shmflg,** specifies the required shared memory flag/s such as IPC_CREAT (creating new segment) or IPC_EXCL (Used with IPC_CREAT to create new segment and the call fails, if the segment already exists)

Valid values for cmd

- **IPC_STAT** − Copies the information of the current values of each member of struct shmid_ds to the passed structure pointed by buf. This command requires read permission to the shared memory segment.
- **IPC_SET** − Sets the user ID, group ID of the owner, permissions, etc. pointed to by structure buf.
- **IPC_RMID** − Marks the segment to be destroyed. The segment is destroyed only after the last process has detached it.
- **IPC_INFO** − Returns the information about the shared memory limits and parameters in the structure pointed by buf.
- **SHM_INFO** − Returns a shm_info structure containing information about the consumed system resources by the shared memory.