Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

Computer Engineering Department & Information Technology Engineering Department

Academic Year: 2021-2022

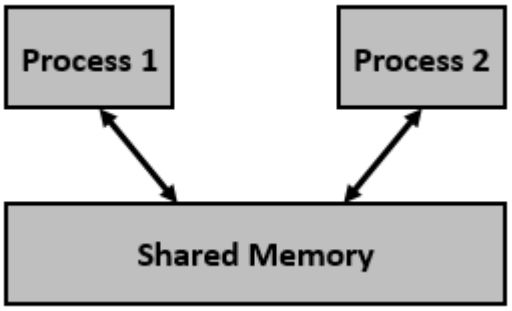
Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

Name	**Pratik Pujari**		
UID no.	**2020300054**	**Class:**	**Comps C Batch**
Experiment No.	5		

AIM:	To simulate a chat application using shared memory
THEORY:	**What is shared memory?** Shared memory is the fastest interprocess communication mechanism. The operating system maps a memory segment in the address space of several processes, so that several processes can read and write in that memory segment without calling operating system functions. However, we need some kind of synchronization between processes that read and write shared memory. Consider what happens when a server process wants to send an HTML file to a client process that resides in the same machine using network mechanisms: • The server must read the file to memory and pass it to the network functions, that copy that memory to the OS's internal memory. • The client uses the network functions to copy the data from the OS's internal memory to its own memory. As we can see, there are two copies, one from memory to the network and another one from the network to memory. And those copies are made using operating system calls that normally are expensive. Shared memory avoids this overhead, but we need to synchronize both processes: • The server maps a shared memory in its address space and also gets access to a synchronization mechanism. The server obtains exclusive access to the memory using the synchronization mechanism and copies the file to memory.

- The client maps the shared memory in its address space. Waits until the server releases the exclusive access and uses the data.

Using shared memory, we can avoid two data copies, but we have to synchronize the access to the shared memory segment.



IPC through shared memory

Inter Process Communication through shared memory is a concept where 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.

The problem with pipes, fifo and message queue – is that for two process to exchange information. The information has to go through the kernel.

- 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). So, 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.

SYSTEM CALLS USED ARE:

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

	• Create the shared memory segment or use an already created shared memory segment (shmget()) • Attach the process to the already created shared memory segment (shmat()) • Detach the process from the already attached shared memory segment (shmdt()) • Control operations on the shared memory segment (shmctl()) ftok(): is use to generate a unique key. shmget(): int shmget(key_t,size_tsize,intshmflg); upon successful completion, shmget() returns an identifier for the shared memory segment. shmat(): Before you can use a shared memory segment, you have to attach yourself to it using shmat(). void *shmat(int shmid ,void *shmaddr ,int shmflg); shmid is shared memory id. shmaddr specifies specific address to use but we should set it to zero and OS will automatically choose the address. shmdt(): When you're done with the shared memory segment, your program should detach itself from it using shmdt(). int shmdt(void *shmaddr); shmctl(): when you detach from shared memory,it is not destroyed. So, to destroy shmctl() is used. shmctl(int shmid,IPC_RMID,NULL);
EXPERIMENT 1	
CODE:	#include <signal.h> #include <stdio.h> #include <stdlib.h> #include <string.h> #include <sys/ipc.h> #include <sys/shm.h> #include <sys/types.h> #include <unistd.h> #define FILLED 0 #define Ready 1

Computer Engineering Department &
Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

```
#define NotReady -1
struct memory {
   char buff[100];
   int status, pid1, pid2;
};
struct memory* shmptr;
void handler(int signum)
{
   if (signum == SIGUSR1) {
      printf("Received User2: ");
      puts(shmptr->buff);
   }
}
int main()
{
   int pid = getpid();
   int shmid;
   int key = 12345;
   shmid = shmget(key, sizeof(struct memory), IPC_CREAT |
0666);
   shmptr = (struct memory*)shmat(shmid, NULL, 0);
   shmptr->pid1 = pid;
   shmptr->status = NotReady;
   signal(SIGUSR1, handler);
   while (1) {
   while (shmptr->status != Ready)
      continue;
      sleep(1);
      printf("User1: ");
      fgets(shmptr->buff, 100, stdin);
      shmptr->status = FILLED;
      kill(shmptr->pid2, SIGUSR2);
   }
   shmdt((void*)shmptr);
   shmctl(shmid, IPC_RMID, NULL);
   return 0;
}
```

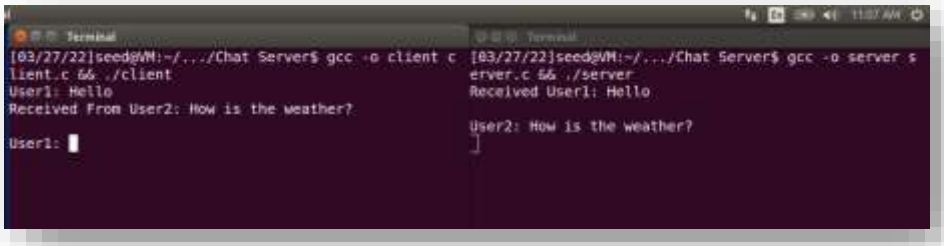
Bharatiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

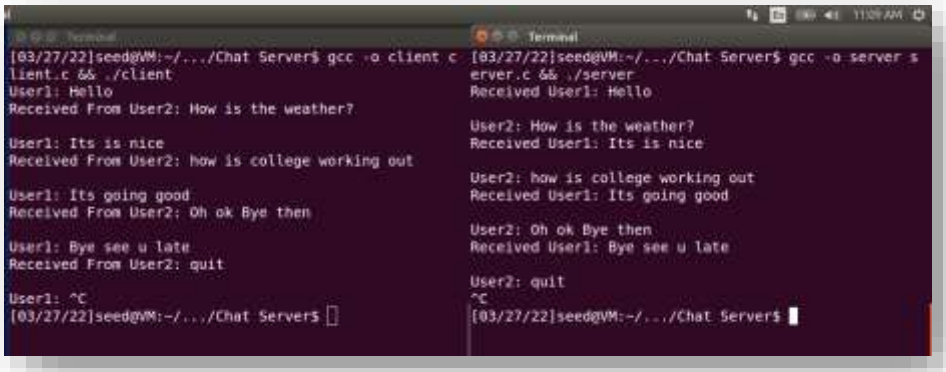
Computer Engineering Department & Information Technology Engineering Department

Academic Year: 2021-2022

Class: S.Y.B.Tech **Sem.:** 4 **Course:** OS

OUTPUT TABLES:





RESULT: I learnt about the shared memory concepts and its function in order to apply the concepts to make a chat application. Learnt how to create two user Id and transferred messages among the two user using shared memory.