##

Experiment No. 5

##

Author: Vidyut Chakrabarti
Semester/section: IV Sem/ B
Roll no.: 68
Date of execution: 19/03/24

##

Aim

To write and execute C programs to demonstrate inter process communication(IPC) using sockets, shared memory and pipes/message queues.

Problem Statement

[1] Write a C program to implement a Chat Service between two users. The Client and the Server Programs may execute on different machines over a local area network (for simplicity you may run them on the same machine). Use TCP sockets.

[2] Write a C program to implement shared memory. The server will receive a number from the client and return the sum-of-its digits / reversed number back to the client.

[3] Write a C program to create bi-directional communication using pipes.

==========

[1]. C program for Client and Server TCP Communication

==========
==========

SERVER.C

==========

```
#include <sys/types.h>
#include <sys/socket.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <netinet/tcp.h>
#include <arpa/inet.h>
#include <unistd.h>

#define SERV_TCP_PORT 9000
```

```
#define MAX_SIZE 100
int main(){
    int sockfd, cl_sockfd, clilen;
    struct sockaddr_in cli_addr, serv_addr;
    int port, len;
    char str[MAX_SIZE];
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0))<0){
           perror("can't open stream socket.\n");
           exit(1);
    }
    bzero((char*)&serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    serv_addr.sin_port = 9000;
    if(bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr))<0){

           perror("can't bind stream socket.\n");
           exit(1);
    }
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    cl_sockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);
    while(1){
        len = recv(cl_sockfd, str, 100, 0);
        printf("Client: %s",str);
        if(strncmp(str, "end" , 3) == 0){
            break;
        }
        printf("\nServer: ");
        fgets(str, 100, stdin);
        len = send(cl_sockfd, str, 100, 0);
    }
    printf("Client Requested Termination.. \n");
    close(cl_sockfd);
    return 0;
 }
```

==========

CLIENT.C

==========

```
#include<string.h>
#include<sys/types.h>
#include <sys/socket.h>
#include<stdlib.h>
#include<strings.h>
#include<stdio.h>
#include<netinet/tcp.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<netdb.h>
```

```
#define _GNU_SOURCE             /* See feature_test_macros(7) */
#define SERV_TCP_PORT 9000
#define MAX_SIZE 100

// int socket(int domain, int type, int protocol);
//int bind(int sockfd, const struct sockaddr *addr,socklen_t addrlen);
//int listen(int sockfd, int backlog);
//int accept4(int sockfd, struct sockaddr *addr,socklen_t *addrlen, int
flags);
// int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
//int connect(int sockfd, const struct sockaddr *addr,socklen_t addrlen);
//ssize_t recv(int sockfd, void *buf, size_t len, int flags);
//int strcmp(const char *s1, const char *s2);
//int strncmp(const char *s1, const char *s2, size_t n);

int main()
{
  int sockfd,cl_sockfd,clilen;
  struct sockaddr_in serv_addr;
  int port,len;
  char str[MAX_SIZE];
  //open a socket stream
  if((sockfd = socket(AF_INET,SOCK_STREAM,0))<0)
  {
  perror("can't open stream socket\n");
  exit(1);
  }
  //initialize sockaddr in structure
  bzero((char*)&serv_addr,sizeof(serv_addr));
  serv_addr.sin_family = AF_INET;
  serv_addr.sin_addr.s_addr = inet_addr("127.0.0.1");
  serv_addr.sin_port = 9000;
  //connect to server
  if(connect(sockfd,(struct sockaddr *)&serv_addr, sizeof(serv_addr))<0)
  {
  perror("can't connect stream socket\n");
  exit(1);}
 //Communicate via system calls
  while(1)
  {
  printf("Client: ");
  fgets(str,100,stdin);
  len = send(sockfd,str,100,0);
  if(strncmp(str,"end",3)==0){
  Break;}
  len =recv(sockfd,str,100,0);
  printf("Server : %s ",str);}
  close(sockfd);
  return 0;
 }
```

[2] C Programs for demonstrating shared memory

SHMWRITER.C

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/shm.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
int i;
void *sharedMem;
int shmid;

/**
shmget allocates a System V shared memory segment
#include <sys/ipc.h>
#include <sys/shm.h>
int shmget (key_t key, size_t size, int shmflg);
**/

shmid = shmget ((key_t) 12345, 1024, 0666 | IPC_CREAT);
printf("Writer: SHMID := %d\n", shmid);

/**
#include <sys/types.h>
#include <sys/shm.h>
void *shmat (int shmid, const void *shmaddr, int shmflg); int shmdt (const
void *shmaddr);
**/

sharedMem = shmat (shmid, NULL, 0);
printf("Writer: Process attached at %p\n", sharedMem);
printf("Enter data: \n");
read(0,sharedMem, 100);
printf("\nNumber Written to Shared Memory..\n");
//shmdt (sharedMem);
return 0;}
```

SHMREADER.C

```
#include <string.h>
#include<stdio.h>
#include <stdlib.h>
```

```
#include <sys/shm.h>
#include <string.h>
#include <sys/types.h>
#include <unistd.h>

int main(){
int i, shmid, len, num, slen, temp;
void *sharedMem;
char buff[100], str[100] = {0};

shmid = shmget ((key_t) 12345, 1024, 0666); //ftok()
printf("Reader: SHMID := %d\n", shmid);

sharedMem = shmat (shmid, NULL, 0);
printf("Reader: Process attached at %p\n", sharedMem);
printf("Data read: %s\n", (char *) sharedMem);

strcpy(str, sharedMem);
slen = strlen(str);

for(i=0;i< slen / 2;i++) {
temp = str[i];
str[i]= str[slen - i - 1];
str[slen- i - 1] = temp;
}
printf("Reveresed number is :%s\n",str);
shmdt (sharedMem);

/*
shmctl System V shared memory control
#include <sys/ipc.h>
#include <sys/shm.h>
int shmctl(int shmid, int cmd, struct shmid ds *buf);
*/

shmctl (shmid, IPC_RMID, NULL);
return 0;
}
```

##

EXECUTION TRACE:

==========

[1] Communication between Client and server

==========
Server
==========

```
vidyut@vidyut-VirtualBox:~/Desktop/B_68/Prac5$ gcc server.c -o server.out
vidyut@vidyut-VirtualBox:~/Desktop/B_68/Prac5$ gcc client.c -o client.out

vidyut@vidyut-VirtualBox:~/Desktop/B_68/Prac5$ ./server.out
Client: Hi, I am Vidyut. Roll no. 68 from Sec B
Server: How may I help you?
Client: This is practical 5 convo using sockets...
Server: Yes, this is the server speaking.
Client: end

Client Requested Termination..

================================================
Client
================================================

vidyut@vidyut-VirtualBox:~/Desktop/B_68/Prac5$ ./client.out
Client: Hi, I am Vidyut. Roll no. 68 from Sec B
Server: How may I help you?
Client: This is practical 5 convo using sockets...
Server: Yes, this is the server speaking.
Client: end

================================================================================
                    [2] Shared memory communication
 ================================================================================
============================
SHMWRITER.C
============================

vidyut@vidyut-VirtualBox:~/Desktop/B_68/Prac5$ ./writer.out
Writer: SHMID := 3
Writer: Process attached at 0x7f88222cb000
Enter data:
99893849290

Number Written to Shared Memory..

============================
SHMREADER.C
============================

vidyut@vidyut-VirtualBox:~/Desktop/B_68/Prac5$ ./reader.out
Reader: SHMID := 3
Reader: Process attached at 0x7f135a3f3000
Data read: 99893849290

Reveresed number is:
09294839899
```