Interprocess Communication (50 points)

Write two programs P1 and P2. The first program P1 needs to generate an array of 50 random strings (of characters) of fixed length each. P1 then sends a group of five consecutive elements of the array of strings to P2 along with the ID's of the strings, where the ID is the index of the array corresponding to the string. The second program P2 needs to accept the received strings, and send back the highest ID received back to P1 to acknowledge the strings received. The program P2 simply prints the ID's and the strings on the console. On receiving the acknowledged packet, P1 sends the next five strings, with the string elements starting from the successor of the acknowledged ID.

We have use IPC mechanism to show above task

1] FIFO 2]. Shared memory 3]. Unix domain sockets

1 **FIFO :**

I used three program to do above task by 1 mkfifo which generate the named fifo

As show:

```
int main(){
    int r;
    r = mkfifo("myfifo",0777);
    printf("created");
return 0;
}
```

After that i used fifo_sender which send the data by using "myfifo" pipe which is above generate First open the pipe to send data than close it to retrieve by fifo_reciver.c and some info like id sent by fifo_reciver.c to notify the fifo_sender.c than code again open" myfifo" pipe to retrieved the id and send it again to receiver file:

```
 res = open("myfifo", O_WRONLY);
        printf("  Data sent at index [%d] is   \t\t:",m);
        for (int i = 0; i < 5; i++) {
            str[i] = alphabate[rand() % 52];
            printf("%c",str[i]);
        }
        printf("\n");
        usleep(100);
        write(res, str, sizeof(str));
        close(res);
```

This similar thing happen by using shared and socket mechanism by IPC

To run the code :

First run sender part than receive part to generate data or transfer the data easily otherwise the process did not work:

Like :

./fifo_sender & ./fifo_reciver

2. Shared data:
Some function i used to generate the shared memory than receiver get this data and send notify as above code did.
Some function are different but working is same:

```
 char buff[1024];
   int shmid;
   shmid = shmget((key_t)1122, 1024, 0666 | IPC_CREAT);
   shared_memory = shmat(shmid, NULL, 0);
   printf("Data is adding to shared_memory....\n");
```

And data is received by shared_receiver.c file as:

```
 strcpy(shared_memory, str[i]);
        sleep(3);
        printf("#%d\tData recived : %s \n",i, (char *)shared_memory);
        i++;
```

Similarly: First run sender part than receive part to generate data or transfer the data easily otherwise the process did not work:
Like :

./share_sender & ./shared_recived

3] socket:
This process make a network and send it to the network or portal and receiver part caught it transfer it to generate data and send data to transmission of data :
First make a portal like this :

```
memset(&server_sockaddr, 0, sizeof(struct sockaddr_un));
    memset(&client_sockaddr, 0, sizeof(struct sockaddr_un));
    memset(buf, 0, 256);
    server_sock = socket(AF_UNIX, SOCK_STREAM, 0);

    server_sockaddr.sun_family = AF_UNIX;
    strcpy(server_sockaddr.sun_path, SOCK_PATH);
    len = sizeof(server_sockaddr);

    unlink(SOCK_PATH);
    rc = bind(server_sock, (struct sockaddr *) &server_sockaddr, len);


    rc = listen(server_sock, backlog);
    printf("socket listening...\n");
```

Receiver part or second code connect to network and transfer the data by client path to server_path:

```
#define SERVER_PATH "sock.server"
```

```c
#define CLIENT_PATH "sock.client"
```

```c
 memset(&server_socket_addr, 0, sizeof(struct sockaddr_un));
    memset(&client_socket_addr, 0, sizeof(struct sockaddr_un));
    client_socket = socket(AF_UNIX, SOCK_STREAM, 0);
    client_socket_addr.sun_family = AF_UNIX;
    strcpy(client_socket_addr.sun_path, CLIENT_PATH);
    len = sizeof(client_socket_addr);
    unlink(CLIENT_PATH);
     bind(client_socket, (struct sockaddr *) &client_socket_addr,
sizeof(client_socket_addr));
    server_socket_addr.sun_family = AF_UNIX;
    strcpy(server_socket_addr.sun_path, SERVER_PATH);
     connect(client_socket, (struct sockaddr *) &server_socket_addr,
sizeof(client_socket_addr));
    char str[50][6] = {{0}};
    for (int i = 0; i < 5; i++){
        for (int j = 0; j < 50; j++){
            str[j][i] = rand() % 26 + 65;

        }


    }
    for (int i =1; i< 50;i++){
        printf("Data sent :%s\n",str[i]);
        send(client_socket, str[i-1], sizeof(str[i-1]), 0);
        sleep(1);
        if (i%5==0&& i>0){
            printf("Index recieved :%d\n",i);
        }
    }
```
Endlly close the client_socket:

```c
 close(client_socket);
    return 0;
}
```

Run:

./socket_sender & ./socket_reciver