INTER PROCESS COMMUNICATIONS

a) FIFO

Server Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>


#define FIFO_FILE "fifo"

int main() {
    int fifo_fd;
    char buffer[256];

    // Creating the FIFO if it doesn't exist
    mkfifo(FIFO_FILE,0666);//This function creates a FIFO (named
pipe) named
    //"myfifo" with read and write permissions for all users.
    //0666 gives read and write permissions to FIFO.

    // Open the FIFO for reading
    fifo_fd = open(FIFO_FILE, O_RDONLY);//Opening FIFO with read
only permission.

    // Read data from the FIFO
    read(fifo_fd, buffer, sizeof(buffer));

    // Displaying received data
    printf("Server Received: %s\n", buffer);

    // Closing the FIFO
    close(fifo_fd);

    // Removing the FIFO
    unlink(FIFO_FILE);
```

```
    return 0;
}
```

Client Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

#define FIFO_FILE "fifo"

int main() {
    int fifo_fd;
    char buffer[256] = "Hello from client!";
    fifo_fd = open(FIFO_FILE, O_WRONLY);//Opening the FIFO to write
    write(fifo_fd, buffer, sizeof(buffer));//Writing into FIFO
    // Closing the FIFO
    close(fifo_fd);
    return 0;
}
```

Output:

```
dhaarani@WINDOWS:~$ gcc fifo_writer.c -o fifo_writer
dhaarani@WINDOWS:~$ ./fifo_writer
dhaarani@WINDOWS:~$
```

```
dhaarani@WINDOWS: ~

dhaarani@WINDOWS:~$ gcc fifo_reader.c -o fifo_reader
dhaarani@WINDOWS:~$ ./fifo_reader
Server Received: Hello from client!
dhaarani@WINDOWS:~$
```

b)Message Queues

Server Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct message {
    long mq_type;
    char mq_text[256];
};

int main() {
    key_t key;
    int mqid;
    key = ftok("msgqfile", 65);//This function generates a unique key
based on a file and an identifier(65).
    mqid = msgget(key, 0666 | IPC_CREAT);//Creates a message queue based
on key generated , if quque exists it opens it
    //0666:read and write permissions.
    struct message msg;
    msgrcv(mqid, &msg, sizeof(msg), 1, 0);//Receives message from the
message queue.
    printf("Server Received: %s\n", msg.mq_text);//Displays received
message.
    // Removing the message queue
    msgctl(mqid, IPC_RMID, NULL);

    return 0;
}
```

Client Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct message {
    long mq_type;
```

```
    char mq_text[256];
};

int main() {
    key_t key;
    int mqid;
    key = ftok("msgqfile", 65);//Generates the same ID as the server
    // Get the message queue ID
    mqid = msgget(key, 0666 | IPC_CREAT);
    struct message msg;
    msg.mq_type = 1;
    sprintf(msg.mq_text, "Shared Memory IPC!");//Sends the content to
mq_text buffer.
    // Send a message
    msgsnd(mqid, &msg, sizeof(msg), 0);//Sends the message to Message
Queue
    return 0;
}
```

Output:

```
dhaarani@WINDOWS:~$ gcc message_queue_writer.c -o message_queue_writer -lrt
dhaarani@WINDOWS:~$ ./message_queue_writer
dhaarani@WINDOWS:~$
```

```
dhaarani@WINDOWS: ~
dhaarani@WINDOWS:~$ gcc message_queue_reader.c -o message_queue_reader -lrt
dhaarani@WINDOWS:~$ ./message_queue_reader
Server Received: Shared Memory IPC!
```

c)Shared Memory:

Server Code:

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

int main() {
    key_t key;
    int shmid;
```

```
    key = ftok("shmfile", 65);//Creates a unique ID based on file and
identifier(65).
    shmid = shmget(key, 1024, 0666 | IPC_CREAT);//Creates a shared memory
segment with 1024B size.
    //0666:Read and write permissions.
    char *shmaddress = (char*) shmat(shmid, (void*)0, 0);// Attaching the
shared memory segment
    printf("Server Received: %s\n", shmaddress);// Displaying the shared
memory data
    shmdt(shmaddress);// Detaching the shared memory segment
    shmctl(shmid, IPC_RMID, NULL); // Remove the shared memory segment
    return 0;
}
```

Client Code:

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

int main() {
    key_t key;
    int shmid;
    key = ftok("shmfile", 65);//Creates a unique ID based on file and
identifier(65).
    shmid = shmget(key, 1024, 0666 | IPC_CREAT);
    char *shmaddress = (char*) shmat(shmid, (void*)0, 0);// Attaching the
shared memory segment
    sprintf(shmaddress, "Shared Memory IPC"); // Writing data to the
shared memory
    shmdt(shmaddress);// Detaching the shared memory segment
    return 0;
}
```

Output:

```
dhaarani@WINDOWS:~$ gcc shared_memory_writer.c -o shared_memory_writer
dhaarani@WINDOWS:~$ ./shared_memory_writer
dhaarani@WINDOWS:~$
```

dhaarani@WINDOWS: ~

```
dhaarani@WINDOWS:~$ gcc shared_memory_reader.c -o shared_memory_reader
dhaarani@WINDOWS:~$ ./shared_memory_reader
Server Received: Shared Memory IPC
dhaarani@WINDOWS:~$
```