# CLIENT:

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

int main()

{

IPC PROJECT CODE

# NAMED PIPES:

int fd;

//FIFO file path

char\* myfifo = "/tmp/myfifo";

//Creating named file FIFO

//mkfifo=make fifo | pathname, permission mkfifo(myfifo, 0666);

char arr1[100], arr2[100]; while(1)

{

program

}

//Open fifo for wirte only fd=open(myfifo, O\_WRONLY); printf("Me: ");

//file descriptor stdin = standard input, where data is sent to and read by the

//fgets = reads a line and store it in the string

//taking input 'arr2' from user, 100 is max length fgets(arr2,100,stdin);

//write the input 'arr2' on FIFO and close it write(fd,arr2,strlen(arr2)+1);

close(fd);

//open fifo for read only fd=open(myfifo, O\_RDONLY);

//read form fifo read(fd,arr1,sizeof(arr1));

//print the read message printf("Kamisha: %s",arr1); close(fd);

return 0;

}

# SERVER:

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

int main()

{

int fd1;

//fifo file path

char\* myfifo = "/tmp/myfifo";

//Creating named file FIFO

//mkfifo=make fifo | pathname, permission mkfifo(myfifo, 0666);

char str1[100], str2[100]; while(1)

{

//open fifo for read only fd1=open(myfifo, O\_RDONLY); read(fd1,str1,100);

//print the read string then close printf("Ushna: %s",str1); close(fd1);

//now open write mode and write. take string from user fd1=open(myfifo, O\_WRONLY);

printf("Me: ");

//taking input 'str2' from user, 100 is max length fgets(str2,100,stdin);

//write the input 'str2' on FIFO and close it write(fd1,str2,strlen(str2)+1);

close(fd1);

}

return 0;

}

# PIPES

#include <sys/types.h> #include <stdio.h> #include <unistd.h> #include <stdlib.h> #include <time.h>

int main()

{

//seed the random number generator with current time so that the numbers will be different every time we run the program

srand(time(NULL));

//creating a pipe, returning '-1' if not created int pipe1[2];

if (pipe(pipe1) == -1) return -1;

//creating child-parent process pid\_t pid = fork();

//parent process if (pid == 0)

{

//initializing a variable to store values in the pipe int parent;

//creating a client text file to write the produced random numbers FILE \*file = fopen("client.txt", "w");

//closing read end of parent in the pipe close(pipe1[0]);

//initializing loop upto 'i' times for (int i = 0; i < 20; i++)

{

//integer parent saving all the random numbers parent = rand();

//writing from the write end of parent in the pipe write(pipe1[1], &parent, sizeof(int));

//writing all the values stored in int parent to the text file we created above fprintf(file, "client (parent) sending value: %d\n", parent);

}

//closing the file and the write end of parent which inserts a EOF value into the pipe fclose(file);

close(pipe1[1]); wait(NULL);

}

//child process else

{

//creating a server text file to read the consumed random numbers that were produced by parent process

FILE \*file = fopen("server.txt", "w");

//initializing a variable to store the values from the pipe int child = 0;

//closing write end of child in the pipe close(pipe1[1]);

//reading from the read end of child in the pipe and executes the loop until read zero

//The function read returns zero when the EOF value has been read from the pipe while(read(pipe1[0], &child, sizeof(int)) != 0)

{

//reading all the values stored in int child and saving it in the server text file fprintf(file, "server (child) receiving value: %d\n", child);

}

//closing the file and the read end of child, which concludes the execution. Inter process communication was done through pipes.

fclose(file); close(pipe1[0]);

}

}

#include <sys/stat.h>

# MESSAGE PASSING

#include <sys/types.h>//data types pid\_t #include <sys/wait.h>//declaration for waiting #include <stdlib.h>//standard libraray functions #include <stdio.h>//standard buffered I/O #include <mqueue.h>//message queues

#include <unistd.h>//define miscellaneous symbolic constants and types #include <fcntl.h>//file control options

#include <time.h>

//defining values to represent two message types #define MSG\_EMPTY 0

#define MSG\_NUMBER 1

void client(mqd\_t queue); void server(mqd\_t queue);

//hold the message type and the integer value typedef struct { char type; int value; } message\_t;

//pass message structs from client process to server process

typedef union { message\_t message; char buffer[sizeof(message\_t)]; } packet\_t;

int main(void) {

//getting and setting the attributes of a msg queue struct mq\_attr attr = {

.mq\_maxmsg = 10,//max no. of msgs

.mq\_msgsize = sizeof(message\_t),//max msg size

.mq\_curmsgs = 0,//no. of msgs currently queued

};

//message queues descriptors to open message queue using mq\_open mqd\_t queue = mq\_open("/mq\_project", O\_CREAT | O\_RDWR, 0666, &attr); srand(time(NULL));

//used for process ids and process group ids. fork to create parent and child process pid\_t pid = fork();

//if parent id > 0 then execute client else execute server if (pid > 0) { client(queue); }

else { server(queue); }

}

void client(mqd\_t queue) {

//passing msg to server process packet\_t packet;

//opening file using fopen

FILE \*f = fopen("client.txt", "w");

for (int i = 0; i < 20; i++)

{

packet.message.type = MSG\_NUMBER; packet.message.value = rand();

mq\_send(queue, packet.buffer, sizeof(message\_t), 0);

fprintf(f, "client (parent) sending value: %d\n", packet.message.value);

}

//loop executed, client sends a final msg of MSG\_EMPTY packet.message.type = MSG\_EMPTY;

mq\_send(queue, packet.buffer, sizeof(message\_t), 0);

//closes the queue and client.txt file mq\_close(queue);

fclose(f); wait(NULL);

}

void server(mqd\_t queue) { packet\_t packet;

FILE \*f = fopen("server.txt", "w");

mq\_receive(queue, packet.buffer, sizeof(message\_t), 0); while (packet.message.type != MSG\_EMPTY)

{

fprintf(f, "server (child) receiving value: %d\n", packet.message.value); mq\_receive(queue, packet.buffer, sizeof(message\_t), 0);

}

mq\_close(queue); fclose(f);

//for destroying msg queue mq\_unlink("/mq\_project");

}