/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*client.c\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
\* **Programmer**:  Sarath Madala   
\*   
\* **Course**:  CSCI 4353.01   
\*   
\* **Date**:  February 26, 2018

\*   
\* **Assignment**:  Program: Client requesting memory from the server  
\*   
\* **Environment**:  UNIX running on putty  
\*   
\* **Files Included**: client.c

\*   
\* **Purpose**:  client sends the structure having a memory request ,private fifo name to which the

\* server writes and the job name to the server. The server returns the no of frames job

\* occupies and starting and ending addresses and fragmentation.

\*   
\* **Input**:  Two integers to which you want find the sum   
\*   
\* **Preconditions**:  None  
\*   
\* **Output**:  Number of frames job occupies, Starting address, Ending address, Fragmentation   
\*   
\* **Postconditions**: None  
\*

\* **Time Taken**: Estimate/Actual

\* Design: 45 mins/ 40 mins

\* Implementation: 100 mins/70mins

\* Testing: 30 mins/ 20mins

\*

\* **Algorithm**:   
\* Initialize struct with an int for memorySize and two char arrays for jobName

\* and privateFIFO respectively

\* input the memorySize and jobName in struct

\* get a privateFIFO name by concating processId to FIFO\_

\* create a private fifo

\* open server fifo and write the struct to it

\* close the sever fifo

\* open private fifo

\* open private fifo of client

\* read the data sent by the client and print

\* close private fifo

\* unlink private fifo

\*            
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <errno.h>

#include <fcntl.h>

/\* Structure for holding clients memory request, job name and private fifo details\*/

struct client{

int memoryRequest;

char jobName[10];

char privateFIFO[10];

}ref;

/\*Structure for holding process details\*/

struct processDetails{

int startingAdd;

int endingAdd;

int numFrames;

int fragmentation;

char error[30];

}buff;

void main(){

int fda; //To Write to server

int fdb; //To read from server

int clientID;

int startingAdd, endingAdd, numFrames, fragmentation;

int i = 0;

printf("\n Enter a Job name: ");

scanf("%s", &ref.jobName); //Name of the Job

printf("\n Enter amount of memory to be requested: ");

scanf("%d", &ref.memoryRequest); //size of memory request

clientID = getpid();

sprintf(ref.privateFIFO, "FIFO\_%d", clientID); //assigning name for private FIFO

if((mkfifo(ref.privateFIFO, 0666) < 0 && errno != EEXIST)){ //Creating private FIFO

perror("Cant create Private fifo");

exit(-1);

}

if((fda = open("FIFO\_to\_server", O\_WRONLY)) < 0) //opening common FIFO

printf("\n Cant open Fifo to Write");

write(fda, &ref, sizeof(ref)); //writing to the server

close(fda); //Closing common FIFO

if((fdb = open(ref.privateFIFO, O\_RDONLY)) < 0) //Opening private FIFO

printf("\n Cant open private fifo to read");

read(fdb, &buff, sizeof(buff)); //Reading from the server

if(buff.error[0] != 0){ //Checking any error message

printf("\n %s \n", buff.error);

exit(-1);

}

/\*Printing process details\*/

printf("\n Starting Address: %d", buff.startingAdd);

printf("\n Ending Address: %d", buff.endingAdd);

printf("\n Number of Frames: %d", buff.numFrames);

printf("\n Fragmentation: %d \n", buff.fragmentation);

close(fdb); //Closing private FIFO

unlink(ref.privateFIFO); //Unlinking private FIFO

}