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

\*   
\* **Assignment**:  Program: Server acting as a paged memory manager

\*   
\* **Environment**:  UNIX running on putty  
\*   
\* **Files Included**: server.c  
\*   
\* **Purpose**:  Server listens for multiple clients and reads the struct sent by the client which has

\* private fifo name, job name, memory request size. The server has to calculate number of

\* frames it took for that job, its starting address, its ending address and internal fragmentation of

\* that job.

\*   
\* **Input**:  two integers for   
\*   
\* **Preconditions**:  None  
\*   
\* **Output**:  A detailed table of main memory   
\*   
\* **Postconditions**: None

\*

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

\* Design: 45 mins/ 40 mins

\* Implementation: 100 mins/120mins

\* Testing: 30 mins/ 30mins  
\*   
\* **Algorithm**:   
\*      Initialize struct like that of client

\* initialize all variables, frameSize = 128

\* determine number of clients and memory size

\*          Create a fifo named FIFO\_to\_server

\* Repeat for every client

\* Open FIFO\_to\_server

\* Read struct from it

\* size = requestSize % frameSize

\* if(size > 0) then and

\* frames = requestSize/frameSize +1

\* fragmentation = frameSize – size

\* end if

\* startingAddress = frames \* frameSize

\* endingAddress = startingAddress + requestSize -1

\* Open private fifo of client

\* write frames, fragmentation, startingAddress, endingAddress to

\* clients privateFIFO

\* Close private fifo

\* Close FIFO\_to\_server

\* End loop

\* Unlink FIFO\_to\_server

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

#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>

#include <string.h>

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

struct server{

int memoryRequest;

char jobName[10];

char privateFIFO[10];

}buf;

/\*Structure for holding process details\*/

struct processDetails{

int startingAdd;

int endingAdd;

int numFrames;

int fragmentation;

char error[30];

}reff;

void main(){

int fda; //To read from client

int fdb; // to write to client

int finish;

int numClients, totalFrames, frameSize = 128, startingFrame = 0;

int lastPage, availableMemory, i, temp = 0;

char \*error1 = "Client requested too small memory"; // Error Message for 0 or -1 memory requests

char \*error2 = "Not enough memory space"; // Error message for insifficient space

char \*error3 = "Too large memory Space"; // Error message for large memory requests

printf("\n How many number of clients: ");

scanf("%d", &numClients); //Number of clients

printf("\n Number of frames in main memory: ");

scanf("%d", &totalFrames); //Number of Frames in main memory

availableMemory = totalFrames \* frameSize; //Available space in main memory

printf("\n Job Name \t Starting Address \t Ending Address \t Number of frames \t fragmentation");

if((mkfifo("FIFO\_to\_server", 0666) < 0 && errno != EEXIST)){ //Creating common FIFO

perror("Cant create FIFO\_to\_server");

exit(-1);

}

for(i=0; i<numClients; i++){

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

printf("Cant open fifo to read");

finish = read(fda, &buf, sizeof(buf)); //Reading from client

memset(&reff.error,0,30);

if(buf.memoryRequest <= 0){ //Validating memory request

//reff.error = error1;

strcpy(reff.error, error1);

temp = 1;

}

if( buf.memoryRequest > (totalFrames \* frameSize)){

strcpy(reff.error, error3);

temp = 1;

}

if(buf.memoryRequest > availableMemory){

//reff.error = error2;

strcpy(reff.error, error2);

temp = 1;

}

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

printf("Cant open private fifo to write");

if(temp == 0){

reff.startingAdd = startingFrame \* frameSize; //Calculating starting address

reff.endingAdd = reff.startingAdd + buf.memoryRequest - 1; //Calculating ending address

lastPage = buf.memoryRequest % frameSize;

reff.numFrames = buf.memoryRequest/frameSize; //calculating number of frames

if(lastPage != 0){

reff.numFrames++;

reff.fragmentation = 128 - lastPage; //Calculating fragmentation

}

else{

reff.fragmentation = 0;

}

printf("\n %s \t\t %d \t\t\t %d \t\t\t %d \t\t\t %d \n", buf.jobName, reff.startingAdd, reff.endingAdd, reff.numFrames, reff.fragmentation);

startingFrame = startingFrame + reff.numFrames; //Determining starting frame

availableMemory = availableMemory - (reff.numFrames \* frameSize);

}

else

temp = 0;

write(fdb, &reff, sizeof(reff)); //Writing to client private fifo

if(finish < 0)

printf("\n Server: error");

close(fdb); //Closing private FIFO

close(fda); //Closing common FIFO

}

unlink("FIFO\_to\_server"); //Unlinking common FIFO

}