SERVER.C

---------

/\*#include<pthread.h>

#include <limits.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <sys/types.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/socket.h>

#include <unistd.h>

#include <sys/wait.h>

#include <time.h>\*/

#include<header.h>

#include<common.h>

#define MAXBUFF 1024

#define PORTNO 3001

pthread\_mutex\_t client\_count\_lock;

int active\_clients = 0;

void\* handleClient(void\* csfdp);

int main()

{

log\_message("INFO","Creating Socket");

int sfd = 0, retValue=0, \*csfd=0;

//int clientAddlen = 0;

struct sockaddr\_in serv\_address, client\_address;

sfd = socket(AF\_INET,SOCK\_STREAM,0);

if(sfd < 0)

{

log\_message("FATAL","Socket creation failed");

perror("socket() ");

exit(EXIT\_FAILURE);

}

printf("\nSocket created with sockfd : %d\n",sfd);

log\_message("INFO","Socket created");

//reset/set address of server

log\_message("INFO","Building Port");

memset(&serv\_address,'\0',sizeof(serv\_address));

serv\_address.sin\_family = AF\_INET;

serv\_address.sin\_port = htons(PORTNO);

serv\_address.sin\_addr.s\_addr = INADDR\_ANY;//inet\_addr("127.0.0.1");

retValue = bind(sfd, (struct sockaddr \*)&serv\_address,sizeof(serv\_address));

if(retValue < 0)

{

log\_message("FATAL","Bind failed");

perror("bind()");

exit(EXIT\_FAILURE);

}

printf("\nBinded the server to the ipaddress with portno\n");

log\_message("INFO","Listening to clients");

retValue = listen(sfd,5);

if(retValue < 0)

{

log\_message("FATAL","Listen Failed");

perror("listen() ");

exit(EXIT\_FAILURE);

}

printf("\nListening to the clients now\n");

pthread\_mutex\_init(&client\_count\_lock, NULL);

while (1) {

pthread\_mutex\_lock(&client\_count\_lock);

if (active\_clients >= 5) {

pthread\_mutex\_unlock(&client\_count\_lock);

log\_message("WARN","Max xlients reached,in waiting state");

sleep(1);

continue;

}

pthread\_mutex\_unlock(&client\_count\_lock);

csfd = malloc(sizeof(int));

if(csfd== NULL)

{

log\_message("FATAL","Memory Allocation Failed");

continue;

}

\*csfd= accept(sfd, NULL, NULL);

printf("\n Client connected");

if (\*csfd== -1) {

log\_message("FATAL","Accept Failed");

free(csfd);

continue;

}

log\_message("INFO","Client connected");

pthread\_mutex\_lock(&client\_count\_lock);

active\_clients++;

pthread\_mutex\_unlock(&client\_count\_lock);

pthread\_t clientThread;

if (pthread\_create(&clientThread, NULL, handleClient, csfd) != 0) {

close(\*csfd);

free(csfd);

log\_message("FATAL","Failed to create thread");

pthread\_mutex\_lock(&client\_count\_lock);

active\_clients--;

pthread\_mutex\_unlock(&client\_count\_lock);

} else {

pthread\_detach(clientThread);

}

}

log\_message("INFO","Closing server socket");

close(sfd);

pthread\_mutex\_destroy(&client\_count\_lock);

return 0;

}

void\* handleClient(void\* csfdp)

{

int csfd= \*(int \*)csfdp;

free(csfdp);

int choice;

int msisdn;

char result[MAXBUFF]= {0,};

static char buffer[MAXBUFF]={0,};

char username[20]={0,};

char password[20]={0,};

while(1)

{

recv(csfd, &choice, sizeof(choice), 0);

switch (choice) {

case 1: // Sign Up

recv(csfd,username,sizeof(username),0);

recv(csfd,password,sizeof(password),0);

int res=process\_signup(csfd,username,password);

if(res==1)

{

send(csfd,"SignUp Succesful, Please Login",50,0);

}

else

send(csfd,"SignUp Failed",30,0);

break;

case 2: // Login

memset(username,0,sizeof(username));

recv(csfd,username,sizeof(username),0);

memset(password,0,sizeof(password));

recv(csfd,password,sizeof(password),0);

res=process\_login(csfd,username,password);

if(res==1)

{

send(csfd,"Login Suuccesful",50,0);

recv(csfd,&choice,sizeof(choice),0);

switch(choice)

{

case 1: // Process CDR

res=process\_cdr\_file();

if(res==1)

send(csfd,"CDR Processed Successfully",50,0);

else

send(csfd,"CDR Processing Failed",30,0);

break;

case 2:// Print/ search

recv(csfd,&choice,sizeof(choice),0);

switch(choice)

{

case 1: // customer billing

recv(csfd,&msisdn,sizeof(msisdn),0);

res=process\_customer\_billing(csfd,msisdn);

if(res==1)

send(csfd,"Customer Billing Successful",50,0);

else

send(csfd,"Customer Billing Unsuccessful",30,0);

break;

case 2: //operator billing

recv(csfd,buffer,sizeof(buffer),0);

res=process\_interoperator\_billing(buffer);

if(res==1)

send(csfd,"Interoperator Billing Successful",50,0);

else

send(csfd,"Interoperator Billing Failed",30,0);

break;

case 3:

log\_message("INFO","Client disconnected");

close(csfd);

pthread\_mutex\_lock(&client\_count\_lock);

active\_clients--;

pthread\_mutex\_unlock(&client\_count\_lock);

pthread\_exit(NULL);

}

}

}

}}

log\_message("INFO","Client disconnected");

close(csfd);

pthread\_mutex\_lock(&client\_count\_lock);

active\_clients--;

pthread\_mutex\_unlock(&client\_count\_lock);

pthread\_exit(NULL);

}