**Lab - 09**

**P2P Network Using IPC**

**Program: MScIT**

**Sem-2**

**Group ID : 28**

**Student Name Student ID**

Dev Adnani 202212012

Saif Saiyed 202212083

**Client.c**

#include <netdb.h>

#include <stdlib.h>

#include <sys/types.h>

#include <stdio.h>

#include <unistd.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/ioctl.h>

#include <netinet/in.h>

#include <net/if.h>

#include <arpa/inet.h>

#include <wait.h>

#include <signal.h>

#define PORT 8080

char server\_ip[20] = "10.0.2.14";

char ip\_address[20];

int readPipe;

int writePipe;

struct req{

char type[1];

char data[1024];

char filename[20];

int save;

};

struct nodeList{

char nodes[5][15];

};

struct interProcComm{

char type[1];

char data[1024];

struct nodeList nl;

int childPID;

};

struct manifest{

char chunk\_IP[5][2][15];

};

void getIP(){

system("ifconfig | grep 'inet ' | sed -n '1 p' | awk '{print $2}' > ip.txt");

FILE \* f = fopen("ip.txt","r");

fgets(ip\_address,15,f);

}

void comm(int sockfd,int pipeSend,int pipeRecv){

sendIP(sockfd);

printf("Sent my IP\n");

int i;

scanf("%d",&i);

struct manifest man;

memset(&man,0,sizeof(struct manifest));

getManifestData(sockfd,&man);

for(int i=0;i<5;i++){

printf("filename :%s IP:%s\n",man.chunk\_IP[i][0],

man.chunk\_IP[i][1]);

}

scanf("%d",&i);

getChunkFiles(sockfd,&man);

}

void sendIP(int sockfd){

struct req req;

memset(&req,0,sizeof(struct req));

req.type[0]='1';

strcpy(req.data,ip\_address);

send(sockfd,&req,sizeof(struct req),0);

}

void gtValidNodes(int sockfd,int pipeSend,int pipeRecv){

struct req req;

struct nodeList nl;

memset(&nl,0,sizeof(nl));

memset(&req,0,sizeof(struct req));

req.type[0]='2';

send(sockfd,&req,sizeof(struct req),0);

recv(sockfd,&nl,sizeof(struct nodeList),0);

for(int i=0;i<5;i++){

printf("IP [%d] : %s\n",i,nl.nodes[i]);

}

sendNodeListToParent(&nl,pipeSend,pipeRecv);

}

void getManifestData(int sockfd,struct manifest \* man){

struct req req;

memset(&req,0,sizeof(struct req));

req.type[0]='3';

send(sockfd,&req,sizeof(struct req),0);

recv(sockfd,man,sizeof(struct manifest),0);

}

void getChunkFiles(int sockfd,struct manifest \* man){

int count=0;

int status;

for(int i=0;i<5;i++){

if(strlen(man->chunk\_IP[i][1])==0)break;

count++;

if((fork())==0){

printf("trying to get file:%s\n",man->chunk\_IP[i][0]);

clientSegmentForChunk(man->chunk\_IP[i][1],man->chunk\_IP[i][0]);

}

wait(&status);

}

}

void sendNodeListToParent(struct nodeList \*nl,int pipeSend,int pipeRecv){

kill(getppid(),SIGUSR1);

struct interProcComm interProcComm;

interProcComm.type[0]='1';

for(int i=0;i<5;i++){

strcpy(interProcComm.nl.nodes[i],nl->nodes[i]);

}

write(pipeSend,&interProcComm,sizeof(struct interProcComm));

exit(0);

}

void clientSegmentForChunk(char \* server\_ip,char \* filename){

int sockfd;

struct sockaddr\_in address;

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

if(sockfd == -1){

printf("Error while creating socket\n");

exit(0);

}

memset(&address,0,sizeof(address));

address.sin\_family = AF\_INET;

address.sin\_port = htons(PORT);

address.sin\_addr.s\_addr = inet\_addr(server\_ip);

if((connect(sockfd,(struct sockaddr\*)&address,sizeof(address)))!=0){

printf("Connection with server failed.\n");

exit(0);

}

printf("Connection with server established\n");

struct req req,req1;

memset(&req,0,sizeof(struct req));

req.type[0]='4';

strcpy(req.filename,filename);

send(sockfd,&req,sizeof(struct req),0);

memset(&req1,0,sizeof(struct req));

recv(sockfd,&req1,sizeof(struct req),0);

FILE \* f = fopen("dev\_saif.txt","a");

printf("-----------\ndata from %s : %s\n---------\n",filename,req1.data);

fprintf(f,"%s",req1.data);

fclose(f);

if(req1.save==1){

char \* temp = "test.p2p";

FILE \* chunk = fopen(temp,"w");

fprintf(chunk,"%s",req1.data);

fclose(chunk);

struct req manReq;

memset(&manReq,0,sizeof(struct req));

manReq.type[0]='5';

strcpy(manReq.filename,filename);

strcpy(manReq.data,ip\_address);

printf("sent req to update manifest data\n");

send(sockfd,&manReq,sizeof(struct req),0);

}

exit(0);

}

void clientSegment(char \* server\_ip,int pipeSend,int pipeRecv){

int sockfd;

struct sockaddr\_in address;

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

if(sockfd == -1){

printf("Error while creating socket\n");

exit(0);

}

memset(&address,0,sizeof(address));

address.sin\_family = AF\_INET;

address.sin\_port = htons(PORT);

address.sin\_addr.s\_addr = inet\_addr(server\_ip);

if((connect(sockfd,(struct sockaddr\*)&address,sizeof(address)))!=0){

printf("Connection with server failed.\n");

exit(0);

}

printf("Connection with server established\n");

comm(sockfd,pipeSend,pipeRecv);

}

void signalHandler(int sig){

struct interProcComm interProcComm;

memset(&interProcComm,0,sizeof(struct interProcComm));

read(readPipe,&interProcComm,sizeof(struct interProcComm));

switch(interProcComm.type[0]){

case '1':startNewConnection(&interProcComm);break;

}

}

void startNewConnection(struct interProcComm \*interProcComm){

strcpy(server\_ip,interProcComm->nl.nodes[0]);

}

int main(){

int pipe1[2];

int pipe2[2];

pipe(pipe1);

pipe(pipe2);

readPipe = pipe1[0];

writePipe = pipe2[0];

int status;

getIP();

printf("My IP: %s",ip\_address);

signal(SIGUSR1,signalHandler);

while(1){

if(fork()==0){

printf("Connecting with server of IP: %s\n",server\_ip);

clientSegment(server\_ip,pipe1[1],pipe2[0]);

}

wait(&status);

}

}

**Server.c**

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

#include <wait.h>

#include <signal.h>

#include <fcntl.h>

#define PORT 8080

char clients[5][20];

int top=1;

int readPipe;

int writePipe;

char ipaddr[20];

struct request{

char type[1];

char data[1024];

char filename[20];

int save;

};

struct clientIP{

int index;

char IP[15];

};

struct nodeList{

char nodes[5][15];

};

struct IPC{

char type[1];

char data[1024];

char IP[15];

char filename[15];

struct clientIP cip;

struct nodeList nl;

};

struct manifest{

char chunk\_IP[5][2][15];

};

struct manifest MAN;

void getIP(){

system("ifconfig | grep 'inet ' | sed -n '2 p' | awk '{print $2}' > serverip.txt");

FILE \* f = fopen("serverip.txt","r");

fgets(ipaddr,15,f);

}

void distributeFile(){

FILE \*f = fopen("data.txt","r");

int count;

char c;

for (c = getc(f); c != EOF; c = getc(f)) count = count + 1;

fclose(f);

int devide = (count / 4)+1;

FILE \* fd = fopen("data.txt","r");

char msg[1024];

for(int i=0; fgets(msg, devide, fd) != NULL ;i++){

if (i==3){

memset(msg,0,1024);

fgets(msg,devide,fd);

}

char filename[20];

sprintf(filename,"chunk%d.p2p",i);

FILE \* nf = fopen(filename,"w");

fprintf(nf,"%s",msg);

strcpy(MAN.chunk\_IP[i][0],filename);

strcpy(MAN.chunk\_IP[i][1],ipaddr);

fclose(nf);

memset(msg,0,1024);

}

for(int i=0;i<5;i++){

printf("filename :%s IP:%s\n",MAN.chunk\_IP[i][0],

MAN.chunk\_IP[i][1]);

}

}

void communicate(int client,int pipeSend,int pipeRecv,int index){

int cont=1;

while(cont == 1){

struct request req;

memset(&req,0,sizeof(struct request));

recv(client,&req,sizeof(struct request),0);

int con = (int)req.type[0];

if(con == 0) continue;

cont = handleClientRequest(&req,client,pipeSend,pipeRecv,index);

}

printf("exited\n");

exit(0);

}

int handleClientRequest(struct request \* req,int client,

int pipeSend,int pipeRecv,int index){

switch(req->type[0]){

case '1':recvClientIP(req,pipeSend,pipeRecv,index);break;

case '2':getClientsIP(client,req,pipeSend,pipeRecv);break;

case '3':sendManifestData(client);break;

case '4':sendChunkFile(client,req);break;

case '5':updateManifest(client,req,pipeSend,pipeRecv);break;

case '6':printf("close connection.\n");return 0;

default:printf("Invalid request from client.\n");

}

return 1;

}

void recvClientIP(struct request \* req,int pipeSend,int pipeRecv,int index){

printf("Connection established with client of IP %s",req->data);

struct flock fl;

fl.l\_type = F\_WRLCK;

fl.l\_whence = SEEK\_SET;

fl.l\_start = 0;

fl.l\_len = 0;

fl.l\_pid = getpid();

fcntl(pipeSend,F\_SETLK,&fl);

kill(getppid(),SIGUSR1);

struct IPC ipc;

memset(&ipc,0,sizeof(struct IPC));

ipc.type[0]='1';

ipc.cip.index = index;

strcpy(ipc.cip.IP,req->data);

write(pipeSend,&ipc,sizeof(struct IPC));

fl.l\_type = F\_UNLCK;

fcntl(pipeSend,F\_SETLK,&fl);

printf("Done\n");

}

void getClientsIP(int client,struct reuqest \* req,int pipeSend,int pipeRecv){

struct flock fl;

fl.l\_type = F\_WRLCK;

fl.l\_whence = SEEK\_SET;

fl.l\_start = 0;

fl.l\_len = 0;

fl.l\_pid = getpid();

fcntl(pipeSend,F\_SETLK,&fl);

kill(getppid(),SIGUSR1);

struct IPC ipc;

memset(&ipc,0,sizeof(struct IPC));

ipc.type[0]='2';

struct nodeList nl;

memset(&nl,0,sizeof(struct nodeList));

write(pipeSend,&ipc,sizeof(struct IPC));

memset(&ipc,0,sizeof(struct IPC));

read(pipeRecv,&nl,sizeof(struct nodeList));

for(int i=0;i<5;i++){

printf("IP [%d] : %s",i,nl.nodes[i]);

}

fl.l\_type = F\_UNLCK;

fcntl(pipeSend,F\_SETLK,&fl);

send(client,&nl,sizeof(struct nodeList),0);

printf("Done2\n");

}

void sendManifestData(int client){

send(client,&MAN,sizeof(struct manifest),0);

}

void sendChunkFile(int client,struct request \* req){

struct request newReq;

char data[1024];

memset(&newReq,0,sizeof(struct request));

FILE \* f = fopen(req->filename,"r");

fgets(data,1024,f);

if(strncmp("chunk0.p2p",req->filename,10) == 0){

newReq.save = 1;

printf("====\nsave\n====\n");

}

strcpy(newReq.data,data);

printf("data: %s\n",newReq.data);

send(client,&newReq,sizeof(struct request),0);

printf("Sent file\n");

fclose(f);

}

void updateManifest(int client,struct request \* req,int pipeSend,int pipeRecv){

printf("Updating manifest for file: %s with IP: %s",req->filename,req->data);

struct flock fl;

fl.l\_type = F\_WRLCK;

fl.l\_whence = SEEK\_SET;

fl.l\_start = 0;

fl.l\_len = 0;

fl.l\_pid = getpid();

fcntl(pipeSend,F\_SETLK,&fl);

kill(getppid(),SIGUSR1);

struct IPC ipc;

memset(&ipc,0,sizeof(struct IPC));

ipc.type[0] = '3';

strcpy(ipc.filename,req->filename);

strcpy(ipc.IP,req->data);

write(pipeSend,&ipc,sizeof(struct IPC));

fl.l\_type = F\_UNLCK;

fcntl(pipeSend,F\_SETFL,&fl);

printf("Manifest Updated\n");

}

void signalHandler(int sig){

struct IPC ipc;

memset(&ipc,0,sizeof(struct IPC));

read(readPipe,&ipc,sizeof(struct IPC));

switch(ipc.type[0]){

case '1':recvIP(&ipc);break;

case '2':sendIPS(&ipc);break;

case '3':recvManifestData(&ipc);break;

}

}

void recvIP(struct IPC \* ipc){

printf("Called the signal: %s\n",ipc->cip.IP);

strcpy(clients[ipc->cip.index],ipc->cip.IP);

displayClientIPS();

}

void sendIPS(struct IPC \* ipc){

printf("Called the signal\n");

struct nodeList nl;

memset(&nl,0,sizeof(struct nodeList));

for(int i=0;i<5;i++){

strcpy(nl.nodes[i],clients[i]);

}

write(writePipe,&nl,sizeof(struct nodeList));

}

void recvManifestData(struct IPC \* ipc){

printf("Called the signal\n");

for(int i=0;i<5;i++){

if( strcmp(MAN.chunk\_IP[i][0],ipc->filename)==0){

strcpy(MAN.chunk\_IP[i][1],ipc->IP);

break;

}

}

}

void displayClientIPS(){

for(int i=0;i<5;i++){

printf("client [%d] : %s\n",i,clients[i]);

}

}

int main(){

int pipes1[2];

int pipes2[2];

pipe(pipes1);

pipe(pipes2);

readPipe = pipes1[0];

writePipe = pipes2[1];

int sockfd,length;

struct sockaddr\_in address;

getIP();

distributeFile();

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

if(sockfd == -1){

printf("Error while creating socket\n");

exit(0);

}

printf("Socket created successfully\n");

memset(&address,0,sizeof(address));

address.sin\_family = AF\_INET;

address.sin\_port = htons(PORT);

address.sin\_addr.s\_addr = htonl(INADDR\_ANY);

if( (bind(sockfd,(struct sockaddr\*)&address,sizeof(address)))!=0){

printf("Error while binding socket.\n");

exit(0);

}

printf("Binded socket\n");

if((listen(sockfd,5))!=0){

printf("Error while listening.\n");

exit(0);

}

printf("Listening...\n");

signal(SIGUSR1,signalHandler);

while(1){

for(int i=0;i<5;i++){

length = sizeof(address);

int client = accept(sockfd,(struct sockaddr\*)&address,&length);

if(fork()==0){

communicate(client,pipes1[1],pipes2[0],i);

}

top++;

}

}

}

**Node.c**

#include <stdio.h>

#include <netdb.h>

#include <netinet/in.h>

#include <stdlib.h>

#include <string.h>

#include <sys/socket.h>

#include <sys/types.h>

#include <unistd.h>

int main(){

int sp1[2];

int sp2[2];

int cp1[2];

int cp2[2];

pipe(sp1);

pipe(sp2);

pipe(cp1);

pipe(cp2);

if(fork()==0){

char spipe1[5];

char spipe2[5];

sprintf(spipe1,"%d",sp1[1]);

sprintf(spipe2,"%d",sp2[0]);

execl("./server.out","server.out",spipe1,spipe2,NULL);

printf("Failed to start server child process\n");

exit(1);

}

int t;

printf("Do you want to start client?");

scanf("%d",&t);

if(fork()==0){

char cpipe1[5];

char cpipe2[5];

sprintf(cpipe1,"%d",cp1[1]);

sprintf(cpipe2,"%d",cp2[0]);

char \*arguments[]={"./client.out",cpipe1,cpipe2,NULL};

execl("./client.out","client.out",cpipe1,cpipe2,NULL);

printf("Failed to start client child process\n");

exit(1);

}

int statusServer,statusClient;

pid\_t pid1,pid2;

pid1 = wait(&statusServer);

pid2 = wait(&statusClient);

}

**Data.txt**

CN Assignment 9

Random data by 202212012

Server Hello ABCD

Random data by 202212083

Server Bye ABCD

**Screenshots:**



