#### COEN 146: Computer Networks

**Lab assignment 3: Programming UDP/IP socket**

**Objectives**

##### To use the copy one binary file to another

##### To develop client/ server applications using UDP/IP socket

##### **Part 1: Copy a binary/ text file to another file**

Problem: Copy a binary/text file to another file.

Analysis:

* Input: pass file names as arguments to main, so your main function needs to be defined as follows:

int main( int argc, char \* argv[])

Your files: src.dat and dest.dat files.

* File reading can be accomplished by using either:
  + Functions: fopen, fwrite, and fread for binary files or fprintf and fscanf for text files
    - FILE \*fopen(const char \*filename, const char \*mode)
    - fwrite( ptr, int size, int n, FILE \*fp ); or fprintf() (for text files)
    - fread( ptr, int size, int n, FILE \*fp ); or fscanf() (for text files)
    - fclose(ptr);

e.g.

FILE \*fp;

fp = fopen(“src.dat”,"r”);

fp = fopen(“dest.dat","w”);

fwrite(&buf,sizeof(buf),1,fp);

fread(&buf,sizeof(buf),1,fp);

fclose(fp);

OR

* + System calls: open, read, write
    - int open (const char\* Path, int flags [, int mode ]);
    - size\_t read (int fd, void\* buf, size\_t cnt);
    - size\_t write (int fd, void\* buf, size\_t cnt);

e.g.:

int fd = open("foo.txt", O\_RDWR);

           int nw = write(fd, buf, strlen(buf));

int nr = read(fd, buf, 40);

close (fd);

You need to include the following libraries:

* + - #include<sys/types.h>
    - #include<sys/stat.h>
    - #include <fcntl.h>

1. Write your C program to copy files (binary and text) using functions, compile, debug, run, and test
2. Write your C program to copy files (binary and text) using system calls, compile, debug, run, and test

Demonstrate your programs to the TA

##### **Part 2: Client – Server with UDP/IP[[1]](#footnote-1)**

The following two programs give the source code of a UDP client and UDP server (you may download the .c files from Camino).

//UDP Client

#include <stdio.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <netdb.h>

#include <string.h>

int main(){

//Declare socket file descriptor. All  Unix  I/O  streams  are  referenced  by  descriptors

int sockfd;

//Declare sending buffer of size 1k bytes

char sbuf[1024];

//Declare server address

struct sockaddr\_in servAddr;

//Converts  domain  names   into  numerical  IP  addresses  via  DNS

struct hostent \*host;

host = (struct hostent \*)gethostbyname("localhost"); //Local host runs the server. You may use "127.0.0.1": loopback IP address

//Open a socket, if successful, returns  a descriptor  associated  with  an endpoint

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("socket");

exit(1);

}

//Set the server address to send using socket addressing structure

servAddr.sin\_family = AF\_INET;

servAddr.sin\_port = htons(5000);

servAddr.sin\_addr = \*((struct in\_addr \*)host->h\_addr);

//Client to send messages to the server continuously using UDP socket

while(1){

printf("Client: Type a message to send to Server\n");

scanf("%s", sbuf);

sendto(sockfd, sbuf, strlen(sbuf), 0, (struct sockaddr \*)&servAddr, sizeof(struct sockaddr));

}

return 0;

}

// UDP Server

#include <stdio.h>

#include <stdlib.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

int main(){

//Declare socket file descriptor. All  Unix  I/O  streams  are  referenced  by  descriptors

int sockfd;

//Declare receiving buffer of size 1k bytes

char rbuf[1024];

//Declare server address to which to bind for receiving messages and client address to fill in sending address

struct sockaddr\_in servAddr, clienAddr;

socklen\_t addrLen = sizeof(struct sockaddr);

//Open a socket, if successful, returns  a descriptor  associated  with  an endpoint

if ((sockfd = socket(AF\_INET, SOCK\_DGRAM, 0)) < 0) {

perror("Failure to setup an endpoint socket");

exit(1);

}

//Setup the server address to bind using socket addressing structure

servAddr.sin\_family = AF\_INET;

servAddr.sin\_port = htons(5000); //Port 5000 is assigned

servAddr.sin\_addr.s\_addr = INADDR\_ANY; //Local IP address of any interface is assigned (generally one interface IP address)

//Set address/port of server endpoint for socket socket descriptor

if ((bind(sockfd, (struct sockaddr \*)&servAddr, sizeof(struct sockaddr))) < 0){

perror("Failure to bind server address to the endpoint socket");

exit(1);

}

//Sever continuously waits for messages from client, then prints incoming messages.

while (1){

printf("Server waiting for messages from client: \n");

int nr = recvfrom(sockfd, rbuf, 1024, 0, (struct sockaddr \*)&clienAddr, &addrLen);

rbuf[nr] = '\0';

printf("Client with IP: %s and Port: %d sent message: %s\n", inet\_ntoa(clienAddr.sin\_addr),ntohs(clienAddr.sin\_port), rbuf);

}

return 0;

}

1. Compile and run. Note: you may use the IP address 127.0.0.1 (loop back IP address) for a local host, i.e. both of your client and server run on the same machine. Demonstrate your program to the TA:
   1. Your client and server on your same machine
   2. Your client and your neighbor’s server IP address
2. Modify your UDP client and UDP server programs so that your UDP client reads from a “src.dat” file used in steps 1 and 2, then sends to your neighbor’s UDP server to copy to “dest.dat” file
3. [Bonus] Use the code uploaded on Camino to demonstrate TCP client-server file transfer in Step 4.

Demonstrate to the TA and upload your source code to Camino.

**Requirements to complete the lab**

1. Show the TA correct execution of the program you wrote for Part 1 and upload source code to Camino.
2. Show the TA correct execution of the program you wrote for Part 2 and upload source code to Camino.

Be sure to retain copies (machine and/or printed) of your source code. You will want these for study purposes and to resolve any grading questions (should they arise)

Please start each program with a descriptive block that includes minimally the following information:

/\*

\* Name: <your name>

\* Date:

\* Title: Lab3 - Part ….

\* Description: This program … <you should

\* complete an appropriate description here.>

\*/

1. http://beej.us/guide/bgnet/ [↑](#footnote-ref-1)