|  |
| --- |
| **TCPClient.c** |
| **/\* fpont 12/99 \*/**  **/\* pont.net \*/**  **/\* tcpClient.c \*/**  **#include <sys/types.h>**  **#include <sys/socket.h>**  **#include <netinet/in.h>**  **#include <arpa/inet.h>**  **#include <netdb.h>**  **#include <stdio.h>**  **#include <unistd.h> /\* close \*/**  **#define SERVER\_PORT 1500**  **#define MAX\_MSG 100**  **int main (int argc, char \*argv[]) {**  **int sd, rc, i;**  **struct sockaddr\_in localAddr, servAddr;**  **struct hostent \*h;**    **if(argc < 3) {**  **printf("usage: %s <server> <data1> <data2> ... <dataN>\n",argv[0]);**  **exit(1);**  **}**  **h = gethostbyname(argv[1]);**  **if(h==NULL) {**  **printf("%s: unknown host '%s'\n",argv[0],argv[1]);**  **exit(1);**  **}**  **servAddr.sin\_family = h->h\_addrtype;**  **memcpy((char \*) &servAddr.sin\_addr.s\_addr, h->h\_addr\_list[0], h->h\_length);**  **servAddr.sin\_port = htons(SERVER\_PORT);**  **/\* create socket \*/**  **sd = socket(AF\_INET, SOCK\_STREAM, 0);**  **if(sd<0) {**  **perror("cannot open socket ");**  **exit(1);**  **}**  **/\* bind any port number \*/**  **localAddr.sin\_family = AF\_INET;**  **localAddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);**  **localAddr.sin\_port = htons(0);**    **rc = bind(sd, (struct sockaddr \*) &localAddr, sizeof(localAddr));**  **if(rc<0) {**  **printf("%s: cannot bind port TCP %u\n",argv[0],SERVER\_PORT);**  **perror("error ");**  **exit(1);**  **}**    **/\* connect to server \*/**  **rc = connect(sd, (struct sockaddr \*) &servAddr, sizeof(servAddr));**  **if(rc<0) {**  **perror("cannot connect ");**  **exit(1);**  **}**  **for(i=2;i<argc;i++) {**    **rc = send(sd, argv[i], strlen(argv[i]) + 1, 0);**    **if(rc<0) {**  **perror("cannot send data ");**  **close(sd);**  **exit(1);**    **}**  **printf("%s: data%u sent (%s)\n",argv[0],i-1,argv[i]);**    **}**  **return 0;**  **}** |
| **TCPServer.c** |
| **/\* fpont 1/00 \*/**  **/\* pont.net \*/**  **/\* tcpServer.c \*/**  **#include <sys/types.h>**  **#include <sys/socket.h>**  **#include <netinet/in.h>**  **#include <arpa/inet.h>**  **#include <netdb.h>**  **#include <stdio.h>**  **#include <unistd.h> /\* close \*/**  **#define SUCCESS 0**  **#define ERROR 1**  **#define END\_LINE 0x0**  **#define SERVER\_PORT 1500**  **#define MAX\_MSG 100**  **/\* function readline \*/**  **int read\_line();**  **int main (int argc, char \*argv[]) {**    **int sd, newSd, cliLen;**  **struct sockaddr\_in cliAddr, servAddr;**  **char line[MAX\_MSG];**  **/\* create socket \*/**  **sd = socket(AF\_INET, SOCK\_STREAM, 0);**  **if(sd<0) {**  **perror("cannot open socket ");**  **return ERROR;**  **}**    **/\* bind server port \*/**  **servAddr.sin\_family = AF\_INET;**  **servAddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);**  **servAddr.sin\_port = htons(SERVER\_PORT);**    **if(bind(sd, (struct sockaddr \*) &servAddr, sizeof(servAddr))<0) {**  **perror("cannot bind port ");**  **return ERROR;**  **}**  **listen(sd,5);**    **while(1) {**  **printf("%s: waiting for data on port TCP %u\n",argv[0],SERVER\_PORT);**  **cliLen = sizeof(cliAddr);**  **newSd = accept(sd, (struct sockaddr \*) &cliAddr, &cliLen);**  **if(newSd<0) {**  **perror("cannot accept connection ");**  **return ERROR;**  **}**    **/\* init line \*/**  **memset(line,0x0,MAX\_MSG);**    **/\* receive segments \*/**  **while(read\_line(newSd,line)!=ERROR) {**    **printf("%s: received from %s:TCP%d : %s\n", argv[0],**  **inet\_ntoa(cliAddr.sin\_addr),**  **ntohs(cliAddr.sin\_port), line);**  **/\* init line \*/**  **memset(line,0x0,MAX\_MSG);**    **} /\* while(read\_line) \*/**    **} /\* while (1) \*/**  **}**  **/\* WARNING WARNING WARNING WARNING WARNING WARNING WARNING \*/**  **/\* this function is experimental.. I don't know yet if it works \*/**  **/\* correctly or not. Use Steven's readline() function to have \*/**  **/\* something robust. \*/**  **/\* WARNING WARNING WARNING WARNING WARNING WARNING WARNING \*/**  **/\* rcv\_line is my function readline(). Data is read from the socket when \*/**  **/\* needed, but not byte after bytes. All the received data is read. \*/**  **/\* This means only one call to recv(), instead of one call for \*/**  **/\* each received byte. \*/**  **/\* You can set END\_CHAR to whatever means endofline for you. (0x0A is \n)\*/**  **/\* read\_lin returns the number of bytes returned in line\_to\_return \*/**  **int read\_line(int newSd, char \*line\_to\_return) {**    **static int rcv\_ptr=0;**  **static char rcv\_msg[MAX\_MSG];**  **static int n;**  **int offset;**  **offset=0;**  **while(1) {**  **if(rcv\_ptr==0) {**  **/\* read data from socket \*/**  **memset(rcv\_msg,0x0,MAX\_MSG); /\* init buffer \*/**  **n = recv(newSd, rcv\_msg, MAX\_MSG, 0); /\* wait for data \*/**  **if (n<0) {**  **perror(" cannot receive data ");**  **return ERROR;**  **} else if (n==0) {**  **printf(" connection closed by client\n");**  **close(newSd);**  **return ERROR;**  **}**  **}**    **/\* if new data read on socket \*/**  **/\* OR \*/**  **/\* if another line is still in buffer \*/**  **/\* copy line into 'line\_to\_return' \*/**  **while(\*(rcv\_msg+rcv\_ptr)!=END\_LINE && rcv\_ptr<n) {**  **memcpy(line\_to\_return+offset,rcv\_msg+rcv\_ptr,1);**  **offset++;**  **rcv\_ptr++;**  **}**    **/\* end of line + end of buffer => return line \*/**  **if(rcv\_ptr==n-1) {**  **/\* set last byte to END\_LINE \*/**  **\*(line\_to\_return+offset)=END\_LINE;**  **rcv\_ptr=0;**  **return ++offset;**  **}**    **/\* end of line but still some data in buffer => return line \*/**  **if(rcv\_ptr <n-1) {**  **/\* set last byte to END\_LINE \*/**  **\*(line\_to\_return+offset)=END\_LINE;**  **rcv\_ptr++;**  **return ++offset;**  **}**  **/\* end of buffer but line is not ended => \*/**  **/\* wait for more data to arrive on socket \*/**  **if(rcv\_ptr == n) {**  **rcv\_ptr = 0;**  **}**    **} /\* while \*/**  **}** |
| **UDPClient.c** |
| **/\* fpont 12/99 \*/**  **/\* pont.net \*/**  **/\* udpClient.c \*/**  **#include <sys/types.h>**  **#include <sys/socket.h>**  **#include <netinet/in.h>**  **#include <arpa/inet.h>**  **#include <netdb.h>**  **#include <stdio.h>**  **#include <unistd.h>**  **#include <string.h> /\* memset() \*/**  **#include <sys/time.h> /\* select() \*/**  **#define REMOTE\_SERVER\_PORT 1500**  **#define MAX\_MSG 100**  **int main(int argc, char \*argv[]) {**    **int sd, rc, i;**  **struct sockaddr\_in cliAddr, remoteServAddr;**  **struct hostent \*h;**  **/\* check command line args \*/**  **if(argc<3) {**  **printf("usage : %s <server> <data1> ... <dataN> \n", argv[0]);**  **exit(1);**  **}**  **/\* get server IP address (no check if input is IP address or DNS name \*/**  **h = gethostbyname(argv[1]);**  **if(h==NULL) {**  **printf("%s: unknown host '%s' \n", argv[0], argv[1]);**  **exit(1);**  **}**  **printf("%s: sending data to '%s' (IP : %s) \n", argv[0], h->h\_name,**  **inet\_ntoa(\*(struct in\_addr \*)h->h\_addr\_list[0]));**  **remoteServAddr.sin\_family = h->h\_addrtype;**  **memcpy((char \*) &remoteServAddr.sin\_addr.s\_addr,**  **h->h\_addr\_list[0], h->h\_length);**  **remoteServAddr.sin\_port = htons(REMOTE\_SERVER\_PORT);**  **/\* socket creation \*/**  **sd = socket(AF\_INET,SOCK\_DGRAM,0);**  **if(sd<0) {**  **printf("%s: cannot open socket \n",argv[0]);**  **exit(1);**  **}**    **/\* bind any port \*/**  **cliAddr.sin\_family = AF\_INET;**  **cliAddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);**  **cliAddr.sin\_port = htons(0);**    **rc = bind(sd, (struct sockaddr \*) &cliAddr, sizeof(cliAddr));**  **if(rc<0) {**  **printf("%s: cannot bind port\n", argv[0]);**  **exit(1);**  **}**  **/\* send data \*/**  **for(i=2;i<argc;i++) {**  **rc = sendto(sd, argv[i], strlen(argv[i])+1, 0,**  **(struct sockaddr \*) &remoteServAddr,**  **sizeof(remoteServAddr));**  **if(rc<0) {**  **printf("%s: cannot send data %d \n",argv[0],i-1);**  **close(sd);**  **exit(1);**  **}**  **}**  **return 1;**  **}** |
| **UDPServer.c** |
| **/\* fpont 12/99 \*/**  **/\* pont.net \*/**  **/\* udpServer.c \*/**  **#include <sys/types.h>**  **#include <sys/socket.h>**  **#include <netinet/in.h>**  **#include <arpa/inet.h>**  **#include <netdb.h>**  **#include <stdio.h>**  **#include <unistd.h> /\* close() \*/**  **#include <string.h> /\* memset() \*/**  **#define LOCAL\_SERVER\_PORT 1500**  **#define MAX\_MSG 100**  **int main(int argc, char \*argv[]) {**    **int sd, rc, n, cliLen;**  **struct sockaddr\_in cliAddr, servAddr;**  **char msg[MAX\_MSG];**  **/\* socket creation \*/**  **sd=socket(AF\_INET, SOCK\_DGRAM, 0);**  **if(sd<0) {**  **printf("%s: cannot open socket \n",argv[0]);**  **exit(1);**  **}**  **/\* bind local server port \*/**  **servAddr.sin\_family = AF\_INET;**  **servAddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);**  **servAddr.sin\_port = htons(LOCAL\_SERVER\_PORT);**  **rc = bind (sd, (struct sockaddr \*) &servAddr,sizeof(servAddr));**  **if(rc<0) {**  **printf("%s: cannot bind port number %d \n",**  **argv[0], LOCAL\_SERVER\_PORT);**  **exit(1);**  **}**  **printf("%s: waiting for data on port UDP %u\n",**  **argv[0],LOCAL\_SERVER\_PORT);**  **/\* server infinite loop \*/**  **while(1) {**    **/\* init buffer \*/**  **memset(msg,0x0,MAX\_MSG);**  **/\* receive message \*/**  **cliLen = sizeof(cliAddr);**  **n = recvfrom(sd, msg, MAX\_MSG, 0,**  **(struct sockaddr \*) &cliAddr, &cliLen);**  **if(n<0) {**  **printf("%s: cannot receive data \n",argv[0]);**  **continue;**  **}**    **/\* print received message \*/**  **printf("%s: from %s:UDP%u : %s \n",**  **argv[0],inet\_ntoa(cliAddr.sin\_addr),**  **ntohs(cliAddr.sin\_port),msg);**    **}/\* end of server infinite loop \*/**  **return 0;**  **}** |