A Quick Guide to Networking Software

by J. Sanguino 3rd Edition: February 2013

IST - Computer Networks and the Internet 2012/2013

Mission Briefing

Welcome to this quick guide in networking programming. You will be given a username and password to access any of the RC lab computers. They are connected to Internet and running Linux.

Your mission, should you decide to accept it, is to complete the tasks that will be presented as you move along the guide. They involve the development of programs that communicate through the Internet.

The tools that you will be using are the basis for the development of network applications over the Internet (web browsers and servers, email, peer-to-peer, remote logins, file transfers ...).

The kind of network applications you will be able to develop, on your own, at the end of this guide, will only be bounded by your imagination.

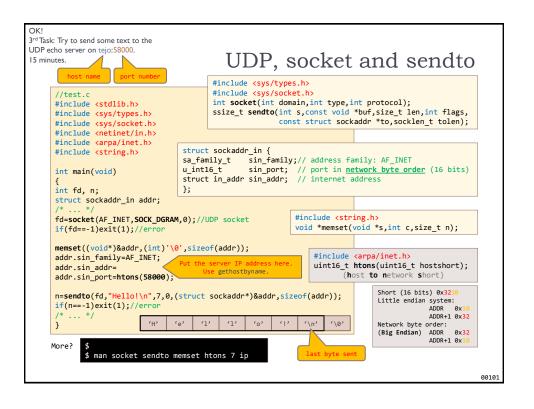
As always, should you or any team member be caught in thrall of network programming, the author would disavow any knowledge of your actions.

00001

```
Login: alunos
Password: alunos
```

```
Welcome, you are inside now.
Ist Task: Get the host name!
You have 10 minutes.
      gethostname
                                                   #include <unistd.h>
                                                   int gethostname(char *name, size_t len);
          //test.c
         #include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
                                                                                 # makefile
                                                                                  \longleftrightarrow gcc test.c -o test
          extern int errno;
          int main(void)
                                                                                 running on tejo.tecnico.ulisboa.pt
          char buffer[128];
                                                                                 gcc test.c -o test
$ ./test
          if(gethostname(buffer,128)==-1)
               printf("error: %s\n",strerror(errno));
                                                                                   ost name: tejo.tecnico.ulisboa.pt
          else printf("host name: %s\n",buffer);
         exit(0);
}
                                     #include <string.h>
                                     char *strerror(int errnum);
                                                                                       $ man gethostname strerror
```

```
Good! Move on!
2<sup>nd</sup> Task: Now that you have a name, get the IP address.
                                                       #include <netdb.h>
      gethostbyname
                                                       struct hostent *gethostbyname(const char *name);
                                             struct hostent{
        //test.c
                                            char *h_name;
char **h aliases;
                                                                    // official host name
       #include <stdio.h>
                                                                   // alias list
        #include <stdlib.h>
                                             int h_addrtype;
                                                                   // host address type
        #include <netdb.h>
                                            int h_length;  // length of address
char **h_addr_list; // list of addresses (NULL term.)
       #include <sys/socket.h>
       #include <netinet/in.h>
       #include <arpa/inet.h>
                                                                          struct in_addr{
       int main(void)
                                                                          uint32_t
                                                                                           s_addr; // 32 bits
                                                                          };
        struct hostent *h;
                                                                                                           \theta x C \theta == 192
        struct in_addr *a;
                                                                                                          \theta_{XA8==168}
                                                                   gcc test.c -o test
        if((h=gethostbyname("tejo"))==NULL)exit(1);//error
                                                                   $ ./test
official host name: tejo.tecnico.ulisboa.pt
internet address: 192.168.0.1 (C0A80001)
       printf("official host name: %s\n",h->h_name);
        a=(struct in_addr*)h->h_addr_list[0];
       printf("internet address: %s (%081X)\n",inet_ntoa(*a),(long unsigned int)ntohl(a->s_addr));
       exit(0);
                      #include <svs/socket.h>
                                                                      #include <arpa/inet.h>
                      #include <netinet/in.h>
                                                                      uint32_t ntohl(uint32_t netlong);
                      #include <arpa/inet.h>
                                                                              (network to host long)
                      char *inet_ntoa(struct in_addr in);
                                                                      Long (32 bits) 0x<mark>76543210</mark>
                                                                      ADDR 0x10
ADDR+1 0x32
                                                                                            Network byte order
                                                                                                ADDR
       More?
               $ man gethostbyname inet_ntoa 7 ip
                                                                                                ADDR+2 0x32
                                                                          ADDR+2 0x54
                                                                          ADDR+3 0x76
                                                                                                ADDR+3 0x10
                                                                                                                      00100
```

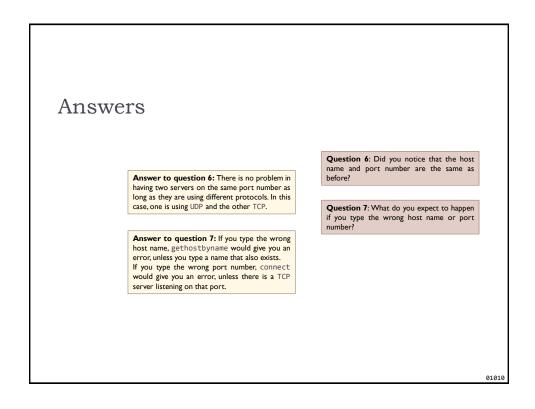


```
4th Task: Now, receive the echo from the UDP echo server.
20 minutes.
                                                               UDP and recyfrom
      #include <stdlib.h>
                                        #include <sys/types.h>
      #include <sys/types.h>
                                        #include <sys/socket.h>
                                        #include <sys/socket.h>
      #include <netinet/in.h>
      #include <arpa/inet.h>
      #include <string.h>
      int main(void)
                                                                 gcc test.c -o test
                                                                   ./test
                                                                 echo: Hello!
      int fd, n, addrlen;
      struct sockaddr in addr;
      char buffer[128];
                                                                               Question 2: How do you know the
      /* ... */// see previous task code
                                                                               message you received came from the
                                                                               UDP echo server on tejo:58000.
      addrlen=sizeof(addr);
      n=recvfrom(fd,buffer,128,0,(struct sockaddr*)&addr,&addrlen);
                                                                               Question 3: Which port number is
      if(n==-1)exit(1);//error
                                                                               your UDP client listening to when it
     write(1,"echo: ",6);//stdout
write(1,buffer,n);
                                                                               is waiting for the echo reply?
                                                                               Question 4: How many bytes do
                                           Question I: What happens if the
                                           messages do not arrive at the destination? Try specifying a wrong port number for the destination
                                                                              you expect to receive from recvfrom?
      close(fd);
      exit(0);
                                           echo server. Did you get an error
                                                                               Question 5: Do you expect buffer
                                           message?
                                                                               to be a NULL terminated string?
            $
$ man recvfrom
     More?
                                                                                                                00110
```

Answer to question 3: The system assigned some unused Answers port in the range 1024 through 5000 when you first called sendto and this is the port recvfrom is listening to. If you want to use a specific port number you have to use bind. More on that later. Answer to question I: No message will be received back at the client and it will block in recvfrom. No error will be detected unless Answer to question 4: In this particular case, you should timeouts are used. expect to receive 7 bytes (see sendto in previous slide). You are using UDP. There are no guarantees that the messages will be delivered at the destination, and the order by which they are Answer to question 5: In this particular case, you should delivered may not be the same in which they not expect buffer to be NULL terminated. See sendto in previous slide and notice that the '\0' was not were transmitted. transmitted Answer to question 2: You have to check the recvfrom addr output argument. See, in the Question 2: How do you know the next slide, how to use gethostbyaddr for message you received came from the UDP echo server on tejo:58000. that purpose. If you only want to receive messages from a specific address, then use send and recv. Find out more on manual page 2 (man 2 send Question 3: Which port number is your UDP client listening to when it recv). is waiting for the echo reply? Question I: What happens if the Question 4: How many bytes do messages do not arrive at the you expect to receive from destination? Try specifying a wrong recvfrom? port number for the destination echo server. Did you get an error Question 5: Do you expect buffer message? to be a NULL terminated string?

```
5<sup>th</sup> Task: Check who sent you the message.
      gethostbyaddr
                                        #include <netdb.h>
                                        #include <sys/socket.h> /* for AF_INET */
        //test.c
                                        struct hostent *gethostbyaddr(const void *addr,int len,int type);
        #include <stdio.h>
        #include <netdb.h>
        #include <sys/socket.h>
                                                                        $ make
                                                                        gcc test.c -o test
        int main(void)
                                                                        echo: Hello!
        int fd, n, addrlen;
struct sockaddr_in addr;
char buffer[128];
                                                                        sent by [tejo.tecnico.ulisboa.pt:58000]
        /* ... */// see previous task code
                                                                             uint16_t ntohs(uint16_t netshort);
                                                                                      (network to host short)
        addrlen=sizeof(addr);
n=recvfrom(fd,buffer,128,0,(struct sockaddr*)&addr,&addrlen);
if(n==-1)exit(1);//error
        h=gethostbyaddr(&addr.sin_addr,sizeof(addr.sin_addr),AF_INET);
        printf("sent by [%s:%hu]\n",inet_ntoa(addr.sin_addr),ntohs(addr.sin_port));
else printf("sent by [%s:%hu]\n",h->h_name,ntohs(addr.sin_port));
                                                                                                                    More?
        exit(0);
                                                                                                $ man gethostbyaddr
                                                                                                                             01000
```

```
OK. Now let's move from UDP to TCP.
TCP is connection-oriented.
6th Task: Connect to the TCP echo server on tejo:58000.
TCP, socket and connect on the TCP echo server on tejo:58000.
TCP is connection-oriented.
                                             #include <sys/types.h>
                                             #include <sys/socket.h>
      #include <stdlib.h>
                                             int connect(int sockfd,const struct sockaddr *serv_addr,
                                                           socklen_t addrlen);
      #include <sys/types.h>
      #include <sys/socket.h>
      #include <netinet/in.h>
      #include <arpa/inet.h>
      #include <string.h>
                                                                        Question 6: Did you notice that the host
                                                                        name and port number are the same as
      int main(void)
                                                                        before?
      int fd, n;
      struct sockaddr_in addr;
                                                                        Question 7: What do you expect to happen
                                                                        if you type the wrong host name or port
      fd=socket(AF_INET,SOCK_STREAM,0);//TCP socket
                                                                        number?
      if(fd==-1)exit(1);//error
      memset((void*)&addr,(int)'\0',sizeof(addr));
      addr.sin_family=AF_INET;
      addr.sin_addr=
                                    As before, put the server IP address here
      addr.sin_port=htons(58000);
      n=connect(fd,(struct sockaddr*)&addr,sizeof(addr));
      if(n==-1)exit(1);//error
                                                                                                        More?
```



```
7^{\text{th}} Task: Send some text over the connection you have just established
and read the response.
10 minutes.
                                                                 TCP, write and read
       //test.c
       #include <unistd.h>
                                                          ssize_t write(int fd,const void *buf,size_t count);
ssize_t read(int fd,void *buf,size_t count);
       #include <string.h>
/* ... */
       int main(void)
       int fd, nbytes, nleft, nwritten, nread; char *ptr, buffer[128];
/* ... */// see previous task code
                                                                                              gcc test.c -o test
$ ./test
                                                         also used to write and
  read to/from files
                                                                                              echo: Hello!
       ptr=strcpy(buffer, "Hello!\n");
       nbytes=7;
       nleft=nbvtes:
       while(nleft>0){nwritten=write(fd,ptr,nleft);
                                                                                  Question 8: Did you notice that you may
                         if(nwritten<=0)exit(1);//error</pre>
                         nleft-=nwritten;
                                                                                 have to call write and read more than
                         ptr+=nwritten;}
                                                                                 once?
       nleft=nbytes; ptr=buffer;
       while(nleft>0){nread=read(fd,ptr,nleft);
                                                                                  Question 9: What do you expect to happen
                         if(nread==-1)exit(1);//error
                                                                                 if your messages do not arrive at the
                         else if(nread==0)break;//closed by peer
                                                                                 destination?
                         nleft-=nread;
                         ptr+=nread;}
       nread=nbytes-nleft;
       close(fd);
                                                                                                                      More?
       write(1,"echo: ",6);//stdout
write(1,buffer,nread);
                                                                                                   $ man 2 write read
       exit(0);
```

Answers

Answer to question 8: There is no guarantee that write would send all the bytes you requested when you called it. Transport layer buffers may be full. However, write returns the number of bytes that were sent (accepted by the transport layer). So, you just have to use this information to make sure everything is sent.

You may also have to call read more that once, since read would return as soon as data is available at the socket. It may happen that, when read returns, there was still data to arrive. Since read returns the number of bytes read from the socket, you just have to use this information to make sure nothing is missing.

Answer to question 9: If the transport layer can not deliver your messages to the destination, the connection will be lost. In some circumstances, this may take a few minutes due to timeouts. If your process is blocked in a read when the connection is lost, then read would return -I and errno would be set to the appropriate error.

If you call write on a lost connection, write would return -I, errno will be set to EPIPE, but the system would raise a SIGPIPE signal and, by default, that would kill your process. See the next slide for a way to deal with the SIGPIPE signal.

Note however that, if the connection is closed, by the peer

Note however that, if the connection is closed, by the peer process, in an orderly fashion, while read is blocking your process, then read would return 0, as a sign of EOF(end-of-file).

Question 8: Did you notice that you may have to call write and read more than once?

Question 9: What do you expect to happen if your messages do not arrive at the destination?

01100

```
Be careful. If the connection is lost and you write to the socket, the system will raise a SIGPIPE signal and, by default, this will kill your process.

8 Task: Protect the application against SIGPIPE signals.

5 minutes.

TCP and the SIGPIPE signal.h>

typedef void (*signal.h>

typedef void (*signal.h>

typedef void (*signal(int signum, signandler_t handler);

int main(void)

{
 void (*old_handler)(int);//interrupt handler

/* ... */

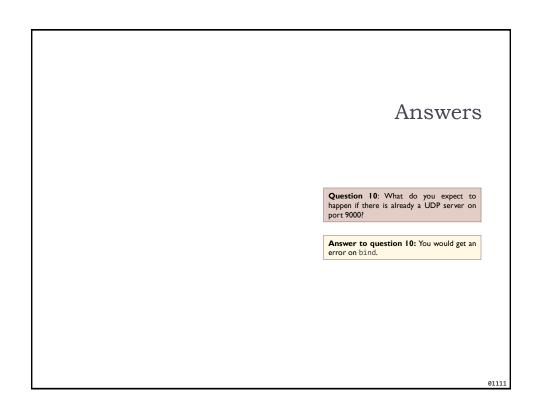
if((old_handler=signal(SIGPIPE,SIG_IGN))==SIG_ERR)exit(1);//error

/* ... */

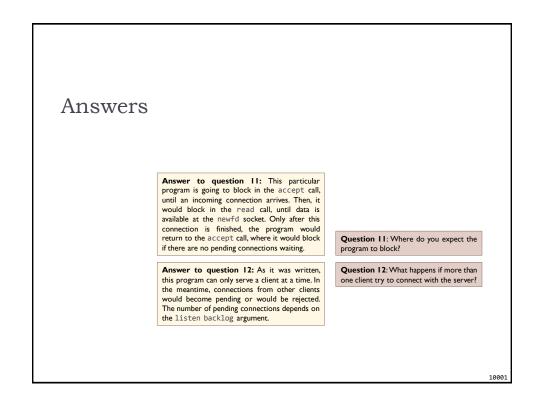
}

Now, if the connection is lost and you write to the socket, the write will return -1 and error will be set to EPIPE.
```

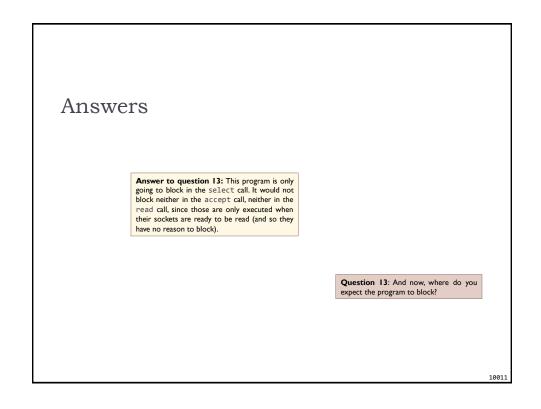
```
Let's move from clients to servers.
Servers have well-known ports.
9th Task: Write a UDP echo server and run it on port 9000.
                                                               UDP server and bind
       #include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
                                                           #include <sys/types.h>
                                                           #include <sys/socket.h>
                                                           int bind(int sockfd,const struct sockaddr *my_addr,
                                                                      socklen_t addrlen);
       int main(void)
                                                                                                                       More?
                                              Use bind to register the server well known address (and port) with the system.
       int fd, addrlen, ret, nread;
                                                                                                           $ man 2 bind
       struct sockaddr_in addr;
       char buffer[128];
                                                                                  Question 10: What do you expect to
       if((fd=socket(AF INET,SOCK DGRAM,0))==-1)exit(1);//error
                                                                                  happen if there is already a UDP server on
                                                                                  port 9000?
       memset((void*)&addr,(int)'\0',sizeof(addr));
addr.sin_family=AF_INET;
                                                                                                    Note: You can also use
       addr.sin_addr.s_addr=htonl(INADDR_ANY);
                                                                                                    bind to register the
       addr.sin_port=htons(9000);
                                                                                                    address (and port) in clients. In that case, if
       ret=bind(fd,(struct sockaddr*)&addr,sizeof(addr));
                                                                                                    you set the port number
       if(ret==-1)exit(1);//error
                                                                                                    to 0, the system assigns
                                                                                                    some unused port in the
       while(1){addrlen=sizeof(addr);
                                                                                                    range 1024 through 5000.
                  nread=recvfrom(fd,buffer,128,0,(struct sockaddr*)&addr,&addrlen);
                  if(nread==-1)exit(1);//error
ret=sendto(fd,buffer,nread,0,(struct sockaddr*)&addr,addrlen);
                  if(ret==-1)exit(1);//error
       //close(fd);
                                                                               uint32_t htonl(uint32_t netlong);
       //exit(0);
}
                                                                                        (host to network long)
                                                                                                                               01110
```

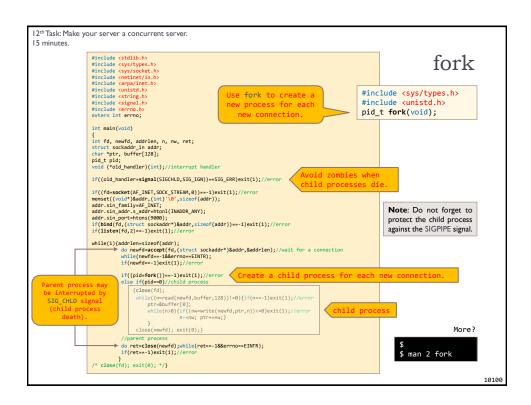


```
Now, do the same, but with TCP.
10th Task: Write a TCP echo server and run it also on port 9000.
20 minutes.
                                    TCP server, bind, listen and accept
        #include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <string.h>
                                       #include <sys/types.h>
#include <sys/socket.h>
                                        int bind(int sockfd,const struct sockaddr *my_addr,
                                                   socklen_t addrlen);
                                       int listen(int sockfd,int backlog);
        int main(void)
                                       int fd, addrlen, newfd;
struct sockaddr_in addr;
        int n, nw;
char *ptr, buffer[128];
        if((fd=socket(AF_INET,SOCK_STREAM,0))==-1)exit(1);//error
        memset((void*)&addr,(int)'\@',sizeof(addr));
addr.sin_family=AF_INET;
addr.sin_addr.s_addr=htonl(INADDR_ANY);
addr.sin_port=htons(9000);
if(bind(fd,(struct sockaddr*)&addr,sizeof(addr))==-1)
                                                                                   associated with the new connection
                                                                                                  Question II: Where do you expect the
                                                                                                  program to block?
            exit(1);//error
        if(listen(fd,5)==-1)exit(1);//error
                                                                                                  Question 12: What happens if more than
                                                                                                  one client try to connect with the server?
        while(1){addrlen=sizeof(addr);
                    if((newfd=accept(fd,(struct sockaddr*)&addr,&addrlen))==-1)
exit(1);//error
                                                                                                  Note: Do not forget to
                    while((n=read(newfd,buffer,128))!=0){if(n==-1)exit(1);//error
                                                                                                  protect your application
                       ptr=&buffer[0];
while(n>0){if((nw=write(newfd,ptr,n))<=0)exit(1);//error</pre>
                                                                                                  against the SIGPIPE signal.
                                                                                                                                        More?
                                     n-=nw; ptr+=nw;}
                    close(newfd);
                                                                                              $ man 2 bind listen accept 7 tcp
        /* close(fd); exit(0); */}
                                                                                                                                                  1000
```



```
If you are already serving a client, send "busy\n" to new incoming clients.
I I th Task: Change the previous code to do that.
15 minutes.
                                                                                                                           select
                                                                #include <sys/time.h>
                                                                #include <sys/types.h>
#include <unistd.h>
                   #define max(A,B) ((A)>=(B)?(A):(B))
                   int main(void)
                                                                int select(int n,fd_set *readfds,fd_set *writefds,
                   fd_set *exceptfds,struct timeval *timeout);
                                                               FD_CLR(int fd,fd_set *set);
FD_ISSET(int fd,fd_set *set);
FD_SET(int fd,fd_set *set);
FD_ZERO(fd_set *set);
                   int marry, connect,
/* ...*/
/* ...*/
/* ...*/
/* febsocket(...); bind(fd,...); listen(fd,...)*/
/* fB_SET(int ff
fD_ZERO(fd_s
state=idle);
while(1)(FD_ZERO(arfds);
maxfd=fd;
if(state=busy)(FD_SET(afd,&rfds);maxfd=max(maxfd,afd);)
                           if(FD_ISSET(fd,&rfds)) fd is ready
                                    {
addrlen=sizeof(addr);
if((newfd-accept(fd,(struct sockaddr*)&addr,&addrlen))==-1)exit(1);//error
switch(state)
                                       }
                            if(FD_ISSET(afd,&rfds)) afd is ready
                                                                                                  Question 13: And now, where do you
                                                                                                 expect the program to block?
                                  else{close(afd); state=idle;}//connection closed by peer
                                                                                                                                       More?
                                                                                                                      $ man 2 select
                                                                                                                                                 10010
```





Further Reading

sanguino@lx.it.pt

Unix Network Programming: Networking APIs: Sockets and XTI (Volume 1), 2^{nd} ed., W. Richard Stevens, 1998, Prentice-Hall PTR, ISBN 013490012X.

Unix Network Programming: Networking APIs: The Sockets Networking API (Volume 1), 3rd ed., W. Richard Stevens, Bill Fenner, Andrew M. Rudoff, 2003, Addison-Wesley Professional, ISBN 0131411551.