A Quick Guide to Networking Software

by J. Sanguino 5th Edition: February 2020

IST - Computer Networks and the Internet 2019/2020

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.

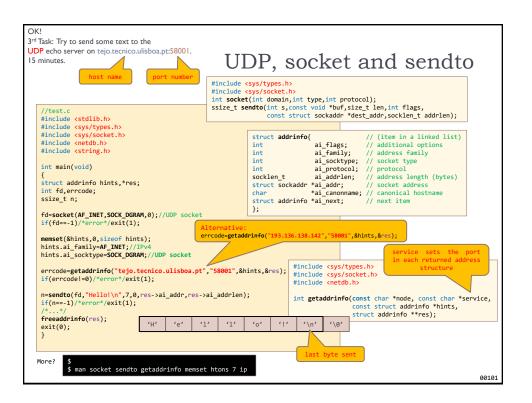
00001

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

```
Welcome, you are inside now.
I^{st} 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>
                                                                                 # makefile

    gcc test.c -o test
tab
          #include <errno.h>
          extern int errno;
          int main(void)
                                                                                 running on tejo.tecnico.ulisboa.pt
          char buffer[128];
          if(gethostname(buffer,128)==-1)
    fprintf(stderr,"error: %s\n",strerror(errno));
                                                                                 gcc test.c -o test
$ ./test
                                                                                  host name: tejo.tecnico.ulisboa.pt
          else printf("host name: %s\n",buffer);
          exit(0);
                                     #include <string.h>
                                      char *strerror(int errnum);
                                                                               More?
                                                                                       $ man gethostname strerror
                                                                                                                           00011
```

```
Good! Move on!
2<sup>nd</sup> Task: Now that you have a name, get the IP address.
                                                                                                     #include <sys/socket.h>
#include <netdb.h>
    getaddrinfo
                                                                                                      int getaddrinfo(const char *node, const char *service,
                                                       #include <arpa/inet.h>
                                                                                                                           const struct addrinfo *hints,
struct addrinfo **res);
       #include <stdio.h:
                                                       const char *inet_ntop(int af,
                                                                                                      void freeaddrinfo(struct addrinfo *res);
      #include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <string.h>
                                                             const void *src,char *dst,
socklen_t size);
                                                                                                      const char *gai_strerror(int errcode);
                                                                                                  struct addrinfo{
                                                                                                                                                (item in a linked list)
                                                                                                                         ai_flags;
                                                                                                                                                 additional options
                                                                                                  int
                                                                                                                        ai_family;
ai_socktype;
                                                                                                 int
                                                                                                                                            // address family
                                                                                                                                                 socket type
       int main(void)
                                                                                                                         ai protocol:
                                                                                                                                                protocol
                                                                                                 socklen_t ai_addrlen;
struct sockaddr *ai_addr;
                                                                                                                                                address length (bytes)
socket address
        truct addrinfo hints,*res,*p;
       int errcode;
char buffer[INET_ADDRSTRLEN];
struct in_addr *addr;
                                                                                                  char
                                                                                                                       *ai_canonname; // canonical hostnam
                                                                                                  struct addrinfo *ai_next;
                                                  #include <string.h>
void *memset(void *s,int c,size_t n);
                                                                                                       hints.ai_family=AF_INET;//IPv4
hints.ai_socktype=SOCK_STREAM;
       hints.ai_flags=AI_CANONNAME;
       if((errcode=getaddrinfo("tejo.tecnico.ulisboa.pt",NULL,&hints,&res))!=0)
            fprintf(stderr, "error: getaddrinfo: %s\n",gai_strerror(errcode));
                                                                                                                                  struct in_addr{
uint32_t s_a
            !\
printf("canonical hostname: %s\n",res->ai_canonname);
                                                                                                                                                     s_addr; // 32 bits
            pint('canonical most make. %s', 'res>al_canonicale'),
for(p=res;p!=NUL;p=p->ai_next){
   addr=&((struct sockaddr_in *)p->ai_addr)->sin_addr;
   printf("internet address: %s (%081X)\n",
                      inet_ntop(p->ai_family,addr,buffer,sizeof buffer),
                                                                                                          uint32_t ntohl(uint32_t netlong);
                                                                                          θxC1==193
                      ({\tt long\ unsigned\ int}) {\tt ntohl}({\tt addr->s\_addr}));
            freeaddrinfo(res);
                                                                                                          Long (32 bits) 0x76543210
Little endian system Ne
                                       gcc test.c -o test
$ ./test
                                                                                                                                           Network byte order
                                        $ ./test
canonical hostname: tejo.tecnico.ulisboa.pt
internet address: 193.136.138.142 (C1888A8E)
                                                                                                                ADDR 0x10
                                                                                                                                                ADDR
                                                                                                                                                         0x76
                                                                                                                ADDR+1 0x32
ADDR+2 0x54
                                                                                                                                                ADDR+1 0x54
ADDR+2 0x32
                                                                                                                ADDR+3 0x76
                                                                                                                                                ADDR+3 0x10
      $ man getaddrinfo inet_ntop memset ntohl 7 ip
                                                                                                                                                                           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 <netdb.h>
      #include <string.h>
      #include <unistd.h>
      int main(void)
                                                                 gcc test.c -o test
                                                                  ./test
      int fd;
                                                                 echo: Hello!
      struct sockaddr in addr:
      socklen_t addrlen;
      ssize t n;
                                                                               Question 2: How do you know the
      char buffer[128];
                                                                               message you received came from the
                                                                               UDP echo server on tejo:58001.
      /*...*///see previous task code
                                                                               Question 3: Which port number is
      addrlen=sizeof(addr);
n=recvfrom(fd,buffer,128,0,(struct sockaddr*)&addr,&addrlen);
                                                                               your UDP client listening to when it
                                                                               is waiting for the echo reply?
      if(n==-1)/*error*/exit(1);
                                                                               Question 4: How many bytes do
      write(1,"echo: ",6);//stdout
                                           Question 1: 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?
      write(1,buffer,n);
      close(fd);
                                           echo server. Did you get an error
      exit(0);
                                                                               Question 5: Do you expect buffer
                                                                               content to be a NULL terminated
                                           message?
            $ man recvfrom
                                                                                                                00110
```

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 sent. 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 getnameinfo for that message you received came from the purpose. UDP echo server on tejo:58001. If you only want to receive messages from a specific address, then use send and recv. Find Question 3: Which port number is out more on manual page 2 (man 2 send 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

port number for the destination echo server. Did you get an error

message?

Answer to question 3: The system assigned some unused

recvfrom?

string?

Question 5: Do you expect buffer

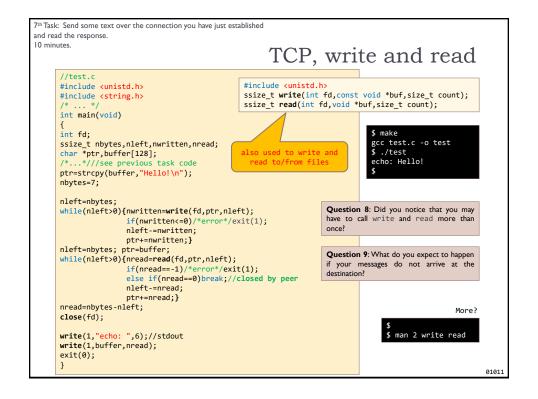
content to be a NULL terminated

00111

```
5th Task: Check who sent you the message.
       getnameinfo
                                                    #include <sys/socket.h>
                                                    #include <netdb.h>
                                                    int getnameinfo(const struct sockaddr *addr, socklen_t addrlen,
         #include <stdio.h>
#include <netdb.h>
#include <sys/socket.h>
/* ... */
                                                                          char *host, socklen_t hostlen,
                                                                          char *serv, socklen_t servlen, int flags);
                                                                                         $ make
         int main(void)
                                                                                         gcc test.c -o test
                                                                                         $ ./test
echo: Hello!
         struct sockaddr_in addr;
socklen_t addrlen;
ssize_t n;
                                                                                         sent by [tejo.tecnico.ulisboa.pt:58001]
               buffer[128];
         int errcode;
         char host[NI_MAXHOST], service[NI_MAXSERV];//consts in <netdb.h>
         /*...*/// see previous task code
         addrlen=sizeof(addr);
n=recvfrom(fd,buffer,128,0,(struct sockaddr*)&addr,&addrlen);
if(n==-1)/*error*/exit(1);
/*...*/
         if((errcode=getnameinfo((struct sockaddr *)&addr.addrlen,host,sizeof host,service,sizeof service,0))!=0)
fprintf(stderr,"error: getnameinfo: %s\n",gai_strerror(errcode));
             printf("sent by [%s:%s]\n",host,service);
                                                                                                                                          More?
         exit(0);
                                                                                                                   $ man getnameinfo
                                                                                                                                                01000
```

```
OK. Now let's move from UDP to TCP.
TCP is connection-oriented.
6^{th} Task: Connect to the TCP echo server on tejo.tecnico.ulisboa:58001.
10 minutes.
                                               TCP, socket and connect
                                              #include <sys/types.h>
                                              #include <sys/socket.h>
      #include <stdlib.h>
                                              int connect(int sockfd,const struct sockaddr *serv_addr,
      #include <sys/types.h>
                                                            socklen_t addrlen);
      #include <sys/socket.h>
#include <netdb.h>
      #include <string.h>
                                                                          Question 6: Did you notice that the host
                                                                          name and port number are the same as
      int main(void)
      struct addrinfo hints,*res;
                                                                          Question 7: What do you expect to happen
      int fd,n;
                                                                          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(&hints,0,sizeof hints);
hints.ai_family=AF_INET;//IPv4
                                                         errcode=getaddrinfo("193.136.138.142","58001",&hints,&res);
      hints.ai_socktype=SOCK_STREAM;//TCP socket
      n=getaddrinfo("tejo.tecnico.ulisboa.pt","58001",&hints,&res);
      if(n!=0)/*error*/exit(1);
      n=connect(fd,res->ai_addr,res->ai_addrlen);
if(n==-1)/*error*/exit(1);
                                                                                                          More?
                                                                                        $ man connect
      }
                                                                                                                   01001
```

Answer to question 6: There is no problem in having two servers on the same port number as long as they are using different protocols. In this case, one is using UDP and the other TCP. Answer to question 7: If you type the wrong host name, get addrinfo would give you an error, unless you type a name that also exists. If you type the wrong port number, connect would give you an error, unless there is a TCP server listening on that port.



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,

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 process, in an orderly fashion, while read is blocking your process, then read would return 0, as a sign of EOF(end-of-file).

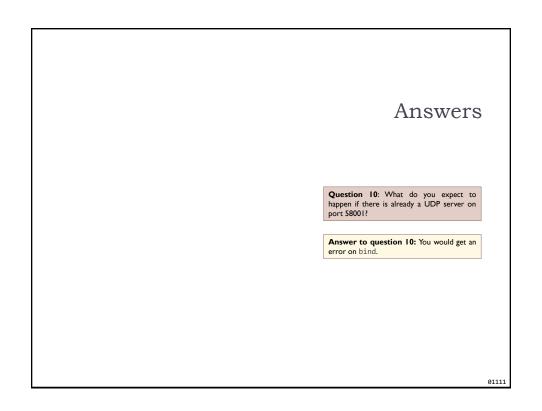
Question 8: Did you notice that you may have to call write and read more than once?

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

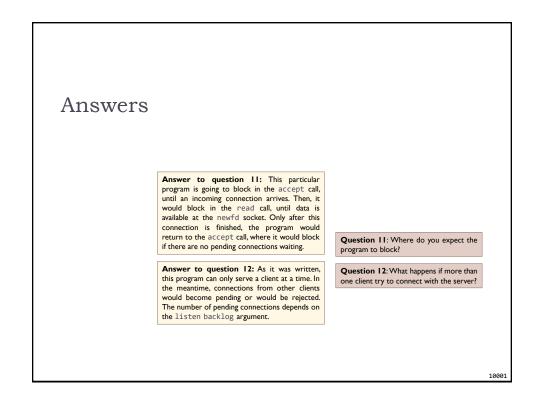
01100

```
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<sup>th</sup> Task: Protect the application against SIGPIPE signals.
    TCP and the SIGPIPE signal
                                               int sigaction(int signum, const struct sigaction *act,
                                                              struct sigaction *oldact);
      #include <signal.h>
      /*···*/
                                                                                                        More?
      int main(void)
                                                                               $ man sigaction 7 signal
      struct sigaction act;
      memset(&act,0,sizeof act);
      act.sa handler=SIG IGN;
      if(sigaction(SIGPIPE,&act,NULL)==-1)/*error*/exit(1);
                          now on, the SIGPIPE
                                                                                                               01101
```

```
Let's move from clients to servers.
Servers have well-known ports.
9th Task: Write a UDP echo server and run it on port 58001.
                                                                                           UDP server and bind
          #include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.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?
          ttuct addrinfo hints,*res;
int fd,errcode;
struct sockaddr_in addr;
socklen_t addrlen;
                                                                 Use bind to register the server well known address (and port) with the system.
                                                                                                                                                        $ man 2 bind
          ssize_t n,nread;
char buffer[128];
                                                                                                                      Question 10: What do you expect to happen if there is already a UDP server on
          if((fd=socket(AF_INET,SOCK_DGRAM,0))==-1)exit(1);//error
                                                                                                                      port 58001?
          memset(&hints,0,sizeof hints);
          immemset(unins);
hints.ai_family=AF_INET;//IPv4
hints.ai_socktype=SOCK_DGRAN;//UDP socket
hints.ai_flags=AI_PASSIVE;
if((errcode=getaddrinfo(NULL, "58001",&hints,&res))!=0)/*error*/exit(1);
                                                                                                                                                Note: You can also use
                                                                                                                                                bind to register the
                                                                                                                                               address (and port) in clients. In that case, if
          if(bind(fd,res->ai_addr,res->ai_addrlen)==-1)/*error*/exit(1);
                                                                                                                                               you set the port number to 0, the system assigns
          while(1){addrlen=sizeof(addr);
    nread=recvfrom(fd,buffer,128,0,(struct sockaddr*)&addr,&addrlen);
    if(nread=-1)/*error*/exit(1);
    n-sendto(fd,buffer,nread,0,(struct sockaddr*)&addr,addrlen);
    if(n==-1)/*error*/exit(1);
                                                                                                                                                some unused port in the
                                                                                                                                                range 1024 through 5000.
          //freeaddrinfo(res);
          //freeaddrinf
//close(fd);
//exit(0);
}
                                                                       Send only the bytes you read
                                                                                                                                                                                      01110
```



```
Now, do the same, but with TCP.
10th Task: Write a TCP echo server and run it also on port 58001.
20 minutes.
                                               TCP server, bind, listen and accept
          #include <stdlib.h>
          #include <std10.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <string.h>
#include <unistd.h>
                                                             #include <sys/types.h>
#include <sys/socket.h>
int bind(int sockfd,const struct sockaddr *my_addr,
                                                            int bind(int sockrd,const struct sockadur *m
socklen_t addrlen);
int listen(int sockfd,int backlog);
int accept(int sockfd,struct sockadur *addr,
socklen_t *addrlen);
           int main(void)
          {
    struct addrinfo hints,*res;
    int fd,newfd,errcode;
    ssize_t n,nw;
    struct sockaddr_in addr;
    char *ptr,buffer[128];
}
          if((fd=socket(AF_INET,SOCK_STREAM,0))==-1)exit(1);//error
          memset(&hints,0,sizeof hints);
hints.ai_family=AF_INET;//IPv4
hints.ai_socktype=SOCK_STREAM;//TCP socket
hints.ai_flags=AI_PASSIVE;
                                                                                                                                                                    Returns a socket
                                                                                                            associated with the new connection
          if((\texttt{errcode} = \textbf{getaddrinfo}(\texttt{NULL}, \texttt{"58001"}, \texttt{\&hints}, \texttt{\&res}))! = \emptyset) / *\texttt{error*} / \texttt{exit}(1);
          if(bind(fd,res->ai_addr,res->ai_addrlen)==-1)/*error*/exit(1);
if(listen(fd,5)==-1)/*error*/exit(1);
address of the connect
                                                                                                                                Question II: Where do you expect the
                                                                                                                                program to block?
          while(1){addrlen=sizeof(addr);
                                                                                                                                Question 12: What happens if more than
                       if((newfd=accept(fd,(struct sockaddr*)&addr,&addrlen))==-1)
    /*error*/exit(1);
                                                                                                                                one client try to connect with the server?
                       Note: Do not forget to
                             while(n>0){if((nw=write(newfd,ptr,n))<=0)/*error*/exit(1);
n-=nw; ptr+=nw;}</pre>
                                                                                                                               protect your application against the SIGPIPE signal.
                        close(newfd);
                                                                                                                                                                                 More?
          //freeaddrinfo(res);close(fd);exit(0);
}
                                                                                                                           $ man 2 bind listen accept 7 tcp
                                                                                                                                                                                              10000
```



```
If you are already serving a client, send "busy\n" to new incoming clients.
11th 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,
                                                    Int main(vas)

Int fd,newfd,afd=0;

fd_set_rfds;

fd_set_rfds;

fd_set_rfds;

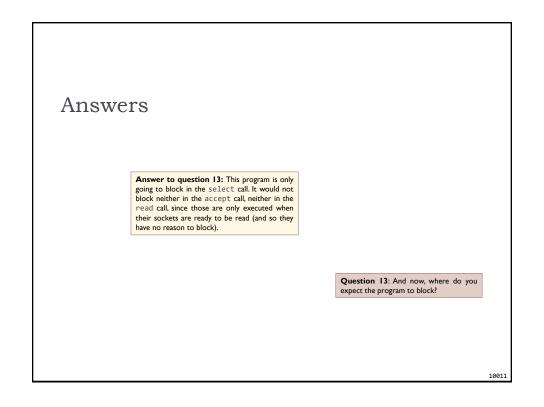
fd_set_rfds;

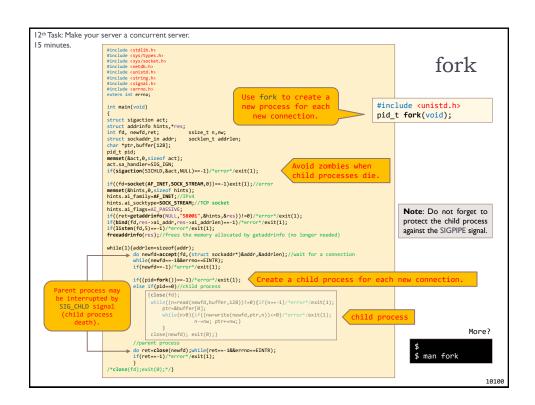
fd_set_rfds;

ff_set_rfds;

ff_set_r
                                                                                                                                                                                                                       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);
                                                                           counter-select(maxfd+1,&rfds,
if(counter<=0)/ferror/exit(1);
if(f0_ISSET(fd,&rfds))</pre>
fd is ready
                                                                                                    }
                                                                            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

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.

sanguino@lx.it.pt 10101