



PADUA UNIVERSITY

ENGINEERING COURSE

Master of Computer Engineering

COMPUTER NETWORKS



Raffaele Di Nardo Di Maio

Indice

1	OSI model	1
2	Application Layer	3
3	C programming	5
3.1	Organization of data	5
3.2	Types of data	5
3.3	Struct organization of memory	5
3.4	Structure of C program	5
4	Network services in C	7
4.1	socket	7
4.2	TCP connection	8
4.2.1	Client	8
4.3	UDP connection	9
5	Shell	11
5.1	Commands	11
5.2	Files	11
5.3	Usefull information about shell	11
5.4	vim	11
5.4.1	.vimrc	11
5.4.2	Shortcuts	12

Capitolo 1

OSI model

Capitolo 2

Application Layer

Capitolo 3

C programming

- 3.1 Organization of data
- 3.2 Types of data
- 3.3 Struct organization of memory
- 3.4 Structure of C program

Capitolo 4

Network services in C

4.1 socket

Entry-point (system call) that allow us to use the network services. It also allows application layer to access to level 4 of IP protocol.

```
#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);\\
```

RETURN VALUE *File Descriptor (FD) of the socket*
-1 if some error occurs and errno is set appropriately
(You can check value of errno including <errno.h>).

domain = *Communication domain*
protocol family which will be used for communication.

AF_INET:	IPv4 Internet Protocol
AF_INET6:	IPv6 Internet Protocol
AF_PACKET:	Low level packet interface

type = *Communication semantics*
SOCK_STREAM: Provides sequenced, reliable, two-way, connection-based bytes stream. An OUT-OF-BAND data mechanism may be supported.
SOCK_DGRAM Supports datagrams (connectionless, unreliable messages of a fixed maximum length).

protocol = *Particular protocol to be used within the socket*
Normally there is only a protocol for each socket type and protocol family (protocol=0), otherwise ID of the protocol you want to use

4.2 TCP connection

In TCP connection, defined by type **SOCK_STREAM** as written in the Section 4.1, there is a client that connects to a server. It uses three primitives (related to File System primitives for management of files on disk) that do these logic actions:

1. start (open bytes stream)
2. add/remove bytes from stream
3. finish (close bytes stream)

TCP is used transferring big files on the network and for example with HTTP, that supports parallel download and upload (FULL-DUPLEX). The length of the stream is defined only at closure of the stream.

4.2.1 Client

The client calls **connect()** function, after **socket()** function of Section 4.1. This function is a system call that client can use to define what is the remote terminal to which he wants to connect.

```
#include <sys/types.h>
#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr *addr, socklen_t addrlen);
```

RETURN VALUE *0* if connection succeeds
 -1 if some error occurs and *errno* is set appropriately

sockfd = *Socket File Descriptor* returned by *socket()*.

addr = *Reference to struct sockaddr*
 sockaddr is a general structure that defines the concept of address.
 In practice it's a union of all the possible specific structures of each protocol.
 This approach is used to leave the function written in a generic way.

addr = *Length of specific data structure used.*

In the following there is the description of struct **sockaddr_in**, that is the specific *sockaddr* structure implemented for family of protocols **AF_INET**:

```
#include <netinet/in.h>

struct sockaddr_in {
    sa_family_t   sin_family; /* address family: AF_INET */
    in_port_t     sin_port;   /* port in network byte order */
    struct in_addr sin_addr;   /* internet address */
};

/* Internet address. */
struct in_addr {
    uint32_t      s_addr;     /* address in network byte order */
};
```

The following piece of code define a structure, used to connect to Google server.

```

1  #include <sys/socket.h>
2  #include <netinet/in.h>
3  #include <netinet/ip.h> /* superset of previous */
4  #include <stdio.h>
5  #include <arpa/inet.h>
6
7  //Identifies the address we want to connect to
8  struct sockaddr_in server;
9
10 int main()
11 {
12     //Creation of socket = number of index in ufdt
13     int s = socket(AF_INET, SOCK_STREAM, 0);
14     char request[100];
15
16     if(s == -1)
17     {
18         perror("Socket Failed");
19         return 1;
20     }
21
22     //Establish the connection
23     unsigned char ip_addr[4] = {216, 58, 208, 163};
24     server.sin_family = AF_INET;
25
26     //Ex. my_htons write a function that change order of bytes (from LE to BE)
27
28     //http service (connection to http service)
29     server.sin_port = htons(80);
30
31     //http service (connection to http service)
32     server.sin_addr.s_addr = *((unsigned int*) ip_addr);
33     int t = connect(s, (struct sockaddr *) &server, sizeof(server));
34
35     if(t == -1)
36     {
37         perror("Connection error");
38         return 1;
39     }
40
41     //Send a request
42     sprintf(request, "GET /\r\n");
43 }

```

Listing 4.1: web_client.c

As mentioned in Section 3.1, network data are organized as Big Endian, so in this case we need to insert the IP address according to this protocol. It can be done as in previous example or with the follow function:

```

#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int inet_aton(const char *cp, struct in_addr *inp);

```

The port number is written according to Big Endian architecture, through the next function:

```

#include <arpa/inet.h>

uint16_t htons(uint16_t hostshort);

```

4.3 UDP connection

UDP connection is defined by type **SOCK_DGRAM** as specified in Section 4.1. It's used for application in which we use small packets and we want immediate feedback directly from application. It isn't reliable because it doesn't need confirmation in transport layer. It's used in Twitter application and in video streaming.

Capitolo 5

Shell

5.1 Commands

strace objFile		List all the system calls used in the program.
gcc -o objFile source -v		List all the path of libraries and headers used in creation of objFile.
netstat	-t	List all the active TCP connections showing domain names.
	-u	List all the active UDP connections showing domain names.
	-n	List all the active, showing IP and port numbers.
nslookup domain		Shows the IP address related to the domain (E.g. IP of www.google.it)

5.2 Files

/etc/services	List all the applications with their port and type of protocol (TCP/UDP).
/usr/include/x86_64-linux-gnu/bits/socket.h	List all the protocol type possible for socket.
/usr/include/x86_64-linux-gnu/sys/socket.h	Definition of struct sockaddr and specific ones.
	.

5.3 vim

5.3.1 .vimrc

In this section there will be shown the file **.vimrc** that can be put in the user home (**~** or **\$HOME** or **-**) or in the path **/usr/share/vim/** to change main settings of the program.

```
1 syntax on
2 set number
3 filetype plugin indent on
4 set tabstop=4
5 set shiftwidth=4
6 set expandtab
7 set t_Co=256
```

Listing 5.1: web_client.c

5.3.2 Shortcuts