Lab Manual
Network Programming Lab

Department of Computer Science & Engineering
College Of Engineering Cherthala , Alappuzha

COLLEGE OF ENGINEERING CHERTHALA, ALAPPUZHA
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision:

To evolve into a center of excellence in the field of computer science and engineering providing innovative and quality engineers contributing to the society and nation

Mission:

To impact high quality professional training with emphasis on state of the art technology in computer science and engineering including professional and ethical values in the young minds

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAMME OUTCOMES (POs)

PO1	Apply the knowledge of mathematics, science and engineering fundamentals to solve complex computer science and engineering related problems.
PO2	Identify, formulate, make literature reviews, and analyze complex computer science and engineering problems to reach substantiated conclusions.
PO3	
PO4	
PO5	
PO6	
PO7	
PO8	
PO9	
PO10	
PO11	
PO12	

Syllabus

***Mandatory List of Exercises/ Experiments**
(Note: At least one program from each topic in the syllabus should be completed in the Lab)

1. Getting started with the basics of network configuration files and networking commands in Linux.*
2. To familiarize and understand the use and functioning of system calls used for network programming in Linux.*
3. Implement client-server communication using socket programming and TCP as transport layer protocol*
4. Implement client-server communication using socket programming and UDP as transport layer protocol*
5. Implementation of a multi user chat server using TCP as transport layer protocol.
6. Implementation of concurrent time server using UDP
7. Simulate sliding window flow control protocols.* (Stop and Wait, Go back N, Selective Repeat ARQ protocols)

8. Implement and simulate algorithm for Distance Vector Routing protocol or Link State Routing protocol.*
9. Implement Simple Mail Transfer Protocol.
10. Implement File Transfer Protocol.*
11. Implement congestion control using a leaky bucket algorithm.*
12. Understanding the Wireshark tool.*
13. Study of NS2 simulator*

Course Outcomes

CO#	**Course Outcomes**
CO1	Use network related commands and configuration files in Linux Operating System. **(Cognitive Knowledge Level: Understand).**
CO2	Develop network application programs and protocols. **(Cognitive Knowledge Level: Apply)**
CO3	Analyze network traffic using network monitoring tools. **(Cognitive Knowledge Level: Apply)**
CO4	Design and set up a network and configure different network protocols. **(Cognitive Knowledge Level: Apply)**
CO5	Develop simulation of fundamental network concepts using a network simulator. **(Cognitive Knowledge Level: Apply)**

Mapping of course outcomes with program outcomes

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PO13
co1	✓	✓	✓					✓		✓		✓	
co2	✓	✓	✓	✓				✓		✓		✓	
co3	✓	✓	✓	✓	✓			✓		✓		✓	
co4	✓	✓	✓	✓	✓	✓		✓		✓		✓	
co5	✓	✓	✓	✓	✓			✓		✓		✓	

Abstract POs defined by National Board of Accreditation

po#	Broad PO	po#	Broad PO
po1	Engineering Knowledge	po7	Environment and Sustainability
po2	Problem Analysis	po8	Ethics
po3	Design/Development of solutions	po9	Individual and teamwork
po4	Conduct investigations of complex problems	po10	Communication
po5	Modern tool usage	po11	Project Management and Finance
po6	The Engineer and Society	po12	Lifelong learning

Reference Books:

1. W. Richard Stevens, Bill Fenner, Andy Rudoff, UNIX Network Programming: Volume 1, The Sockets Networking API, 3rd Edition, Pearson, 2015
2. Lisa Bock, Learn Wireshark: Confidently navigate the Wireshark interface and solve real-world networking problems, Packt Publishing, 2019
3. Teerawat Issariyakul, Ekram Hossain, Introduction to Network Simulator NS2,2nd Edition, Springer,2019

Experiment 1

Getting started with Basics of Network configurations files and Networking

Commands in Linux.

The important network configuration files in Linux operating systems are

1. /etc/hosts

This file is used to resolve hostnames on small networks with no DNS server. This text file contains a mapping of an IP address to the corresponding host name in each line. This file also contains a line specifying the IP address of the loopback device i.e, 127.0.0.1 is mapped to localhost.

A typical hosts file is as shown

127.0.0.1 localhost
127.0.1.1 anil-300E4Z-300E5Z-300E7Z

2. /etc/resolv.conf

This configuration file contains the IP addresses of DNS servers and the search domain.
A sample file is shown
DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 127.0.1.1

3. /etc/sysconfig/network

This configuration file specifies routing and host information for all network interfaces. It contains directives that are global specific. For example if NETWORKING=yes, then /etc/init.d/network activates network devices.

4. /etc/nsswitch.conf

This file includes database search entries. The directive specifies which database is to be searched first.

The important Linux networking commands are

1. ifconfig

This command gives the configuration of all interfaces in the system.
It can be run with an interface name to get the details of the interface.
ifconfig wlan0
Link encap:Ethernet HWaddr b8:03:05:ad:6b:23
inet addr:192.168.43.15 Bcast:192.168.43.255 Mask:255.255.255.0
inet6 addr: 2405:204:d206:d3b1:ba03:5ff:fead:6b23/64 Scope:Global
inet6 addr: fe80::ba03:5ff:fead:6b23/64 Scope:Link
inet6 addr: 2405:204:d206:d3b1:21ee:5665:de59:bd4e/64 Scope:Global
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:827087 errors:0 dropped:0 overruns:0 frame:0
TX packets:433391 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:1117797710 (1.1 GB) TX bytes:53252386 (53.2 MB)

This gives the IP address, subnet mask, and broadcast address of the wireless LAN adapter.
Also tells that it can support multicasting.

If eth0 is given as the parameter, the command gives the details of the Ethernet adapter.

2. netstat

This command gives network status information.
Netstat -i

Iface	MTU	Met	RX-OK	RX-ERR	RX-DRP	RX-OVR	TX-OK	TX-ERR	TX-DRP	TX-OVR	Flg
eth0	1500	0	0	0	0	0	0	0	0	0	BMU
lo	65536	0	12166	0	0	0	12166	0	0	0	LRU
wlan0	1500	0	827946	0	0	0	434246	0	0	0	BMRU

As shown above, the command with -i flag provides information on the interfaces. lo stands for loopback interface.

3. ping

This is the most commonly used command for checking connectivity.

```
ping www.google.com
PING www.google.com (172.217.163.36) 56(84) bytes of data.
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=1 ttl=53 time=51.4 ms
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=2 ttl=53 time=50.3 ms
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=3 ttl=53 time=48.5 ms
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=4 ttl=53 time=59.8 ms
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=5 ttl=53 time=57.8 ms
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=6 ttl=53 time=59.2 ms
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=7 ttl=53 time=68.2 ms
64 bytes from maa05s01-in-f4.1e100.net (172.217.163.36): icmp_seq=8 ttl=53 time=58.8 ms
^C
--- www.google.com ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7004ms
rtt min/avg/max/mdev = 48.533/56.804/68.266/6.030 ms
```

A healthy connection is determined by a steady stream of replies with consistent times. Packet loss is shown by discontinuity of sequence numbers. Large scale packet loss indicates problem along the path.

Experiment 2

To familiarize and understand the use and functioning of System Calls used for Operating system and network programming in Linux.

Some system calls of Linux operating systems

1. Ps

This command tells which all processes are running on the system when ps runs.

ps -ef

UID	PID	PPID	C	STIME TTY	TIME	CMD
root	1	0	0	13:55 ?	00:00:01	/sbin/init
root	2	0	0	13:55 ?	00:00:00	[kthreadd]
root	3	2	0	13:55 ?	00:00:00	[ksoftirqd/0]
root	4	2	0	13:55 ?	00:00:01	[kworker/0:0]
Root	5	2	0	13:55 ?	00:00:00	[kworker/0:0H]
root	7	2	0	13:55 ?	00:00:00	[rcu_sched]
root	8	2	0	13:55 ?	00:00:00	[rcuos/0]

This command gives processes running on the system, the owners of the processes and the names of the processes. The above result is an abridged version of the output.

2. fork

This system call is used to create a new process. When a process makes a fork system call, a new a process is created which is identical to the process creating it. The process which calls fork is called the parent process and the process that is created is called the child process. The child and parent processes are identical, i.e, the child gets a copy of the parent's data space, heap and stack, but have different physical address spaces. Both processes start execution from the line next to the fork. Fork returns the process id of the child in the parent process and returns 0 in the child process.

```
#include<stdio.h>
void main()
{
int pid;
pid = fork();
if(pid > 0)
{
printf (" Iam parent\n");
}
else
{
printf("Iam child\n");
}
}
```

The parent process prints the first statement and the child prints the next statement.

3. exec

New programs can be run using exec system calls. When a process calls exec, the process is completely replaced by the new program. The new program starts executing from its main function.

A new process is not created, process id remains the same, and the current process's text, data, heap, and stack segments are replaced by the new program. exec has many flavors one of which is execv.

execv takes two parameters. The first is the pathname of the program that is going to be executed. The second is a pointer to an array of pointers that hold the addresses of arguments. These arguments are the command line arguments for the new program.

4. wait

When a process terminates, its parent should receive some information regarding the process like the process id, the termination status, amount of CPU time taken etc. This is possible only if the parent process waits for the termination of the child process. This waiting is done by calling the wait system call. When the child process is running, the parent blocks when wait is called. If the child terminates normally or abnormally, wait immediately returns with the termination status of the child. The wait system call takes a parameter which is a pointer to a location in which the termination status is stored.

5. Exit

When exit function is called, the process undergoes a normal termination.

6. open

This system call is used to open a file whose pathname is given as the first parameter of the function. The second parameter gives the options that tell the way in which the file can be used.

open(filepathname , O_RDWR);

This causes the file to be read or written. The function returns the file descriptor of the file.

7. read

This system call is used to read data from an open file.

read(fd, buffer, sizeof(buffer));

The above function reads sizeof(buffer) bytes into the array named buffer. If the end of file is encountered, 0 is returned, else the number of bytes read is returned.

8. write

Data is written to an open file using write function.

write(fd, buffer, sizeof(buffer));

System calls for network programming in Linux

1. Creating a socket

int socket (**int** domain, **int** type, **int** protocol);

This system call creates a socket and returns a socket descriptor. The domain parameter specifies a communication domain; this selects the protocol family which will be used for communication. These families are defined in <sys/socket.h>. In this program the AF_INET family is used. The type parameter indicates the communication semantics. SOCK_STREAM is used for tcp connection while SOCK_DGRAM is used for udp connection. The protocol parameter specifies the protocol used and is always 0. The header files used are <sys/types.h> and <sys/socket.h>.

Experiment 3

Implementation of Client-Server communication using Socket Programming and TCP as transport layer protocol

Aim: Client sends a string to the server using tcp protocol. The server reverses the string and returns it to the client, which then displays the reversed string.

Description:

Steps for creating a TCP connection by a client are:

1. Creation of client socket

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

This function call creates a socket and returns a socket descriptor. The domain parameter specifies a communication domain; this selects the protocol family which will be used for communication. These families are defined in <sys/socket.h>. In this program, the domain AF_INET is used. The socket has the indicated type, which specifies the communication semantics. SOCK_STREAM type provides sequenced, reliable, two-way, connection based byte streams. The protocol field specifies the protocol used. We always use 0. If the system call is a failure, a -1 is returned. The header files used are sys/types.h and sys/socket.h.

2. Filling the fields of the server address structure.

The socket address structure is of type struct sockaddr_in.

```
struct sockaddr_in {

u_short sin_family;
u_short sin_port;
struct in_addr sin_addr;
char sin_zero[8]; /*unused, always zero*/
};
struct in_addr {

u_long s_addr;

};
```

The fields of the socket address structure are
sin_family which in our case is AF_INET
sin_port which is the port number where socket binds
sin_addr which is the IP address of the server machine

The header file that is to be used is **netinet/in.h**

struct sockaddr_in servaddr;
servaddr.sin_family = AF_INET;
servaddr.sin_port = htons(port_number);

Why htons is used ?. Numbers on different machines may be represented differently (big-endian machines and little-endian machines). In a little-endian machine the low order byte of an integer appears at the lower address; in a big-endian machine instead the low order byte appears at the higher address. Network order, the order in which numbers are sent on the internet is big-endian.