Expt. No. 1 Date

Page No. 1

Network configuration files and Networking commands in linux

Aim

Getting started with the basics of network configuration files and Networking commands in Linux.

The important Linux Networking commands are

(i) ifconfig :- This command gives the configuration of all interfaces in the system. It can be run with an interfaces name to get the details of the interface.
eg:- ifconfig → This command gives the IP address and broadcast address of the wireless lan adapter.

(ii) netstat :- This command give network status information
eg: netstat -i → This command with -i flag provides information on the interface.

(iii) ping :- This is the most commonly used command for checking connectivity.

Important Network configuration files in Linux operating system

(i) /etc/hosts :- This file is used to resolve host names on small network with no dns server. This text file contains a mapping of an IP address to the corresponding hosting name in each lin.

Teacher's Signature

Date

Expt. No. Page No. 2

(ii) /etc/resolv.conf :- This configuration file contains the ipaddress of dns server and the search domain.

(iii) /etc/sysconfig/network:- This configuration file specifies routing and host information for all network interface.

(iv) /etc/nsswitch.conf :- This file includes database search entries.

Result.

studies the basics of networking configuration files and networking commands in Linux.

Teacher's Signature

Expt. No. 2 Page No. 3 Date

system Calls in operating System

Aim

To familiarize and understand the use and functioning of systemcalls used for operating system.

(i) ps:- The command gives process running on the system, the owners of the processes and the names of the processes.

(ii) Fork:- This system call is used to create new process. when a process makes a Fork system call a new 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.

(iii) exec:- New programs can be run using exec system call. when a process calls exec, the process is completely replaced by the new program. The new program starts executing from its main function.

(iv) wait:- when a process terminates its parents, should receive some information regarding the process like the process id, 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.

Teacher's Signature

Expt. No.

(v) Exit : when exit - function is called - the process undergoes a normal termination.

(vi) open :- This system call is used to open a file whose path name is given as the first parameter of the function. The second parameters gives the options that tell the way in which the file can be used.
eg:- open(file pathname, O_RDWR);
This causes the file to be read / written.

(vii) read :- This system call is used to read data from an open file.
read (fd buffer, sizeof(buffer));

(viii) write :- Data is written to an open file using write function.
write (fd, buffer, sizeof(buffer));

Result:
Familarized and understood the use and functioning of system calls in OS.

Teacher's Signature _____

Expt. No. 3

Page No. 5

Implementation of client-server communication using socket programming and TCP as transport layer protocal.

Aim: To implement client-server communication using socket programming and TCP as transport layer protocal.

Algorithm

client

1. create socket
2. connect the socket to the server.
3. Read the string to be sent to the server using socket.
4. Read the string from the socket and display it on the standard output.
5. close the socket

server

1. create listening socket.
2. Bind IP address and port number to the socket.
3. Listen for incoming requests on the listening socket.
4. Accept the incoming request.
5. connection socket is created when accept returns.
6. Read the string using the connection socket from the client.
7. send the string to the client using the connection socket
8. close the connection socket
9. close the listening socket.
10. stop

Teacher's Signature

Expt. No. _____ Page No. 6

Date _____

Result.

implement client-server communication using socket programming and TCP as transport layer protocol.

Teacher's Signature _____

Expt. No. 4. Page No. 7

implementation of client-server communication using socket programming and UDP as transport layer protocol

Aim
To implement client-server communication using socket programming and UDP as transport layer protocol.

Algorithm :-

UDP client.

1. create a UDP socket.
2. send a message to the server.
3. wait until response from the server is received.
4. process reply and go back to step 2 if necessary.
5. close socket descriptor and exit.

UDP server.

1. create a UDP socket
2. Bind the socket to the server address.
3. wait until the datagram packet arrives from the client.
4. process the datagram packet and send a reply to the client.
5. Go back to step 3.

Teacher's Signature _____

Date _____

Expt. No. _____ Page No. 8

Result.

implement client-server communication using socket programing and UDP as transport layer protocol.

Teacher's Signature _____

Expt. No. 5 Page No. 9 Date

Implement a concurrent time server using UDP as transport layer protocol by executing the program at remote server.

Aim: To implement a concurrent time server using UDP as transport layer protocol by executing the program at remote servers. client sends a time request to server and server sends its system time back to the client. client displays the result.

Algorithm.

UDP client.

1. create a socket for UDP using the function call, socket(AF_INET, SOCK_DGRAM, 0);
2. The bzero() function places null bytes of memory area pointed to by local. bzero((char *) &local, sizeof(local));
3. initialize the structure sockaddr_in members of sin_family, sin_addr, sin_port
4. Bind the socket to its port using bind(s, (struct sockaddr *) &local, sizeof(local)
5. The bzero() function places null bytes of memory area pointed to by servaddr. bzero((char *) &servaddr, sizeof(local));
6. client sends TIME request to server using sendto(s, buffer, sizeof(buffer), 0, (struct sockaddr*) &servaddr, sizeof(servaddr)
7. client receives TIME response from server using recvfrom() function as follows:
   recvfrom(s, buffer, 1024, 0, (struct sockaddr *) &servaddr, &t)
8. Prints the received message in client's terminal.

Teacher's Signature

Date

Expt. No. Page No. 10

UDP Server

1. create a socket for UDP using the function call, socket (AF_INET, sock_DGRAM, 0);
2. Declare a time object variable of of data type, time_t.
3. The bzero() function places null bytes of the memory area pointed to by local. bzero ((char *)& servaddr, sizeof(servaddr));
4. Initialize the structure sockaddr_in members of sin_family, sin_addr, sin_port.
5. Bind the socket to its port using bind (s, (struct sockaddr*) & servaddr, sizeof(servaddr)
6. Receive time request from client using recvfrom(s, buffer, 1024, 0, (struct sockaddr*) & cliaddr, &t
7. Initializes ct = time (NULL) and prints the current date & time by calling ctime (&ct).
8. child process is created. Parent process stops listening for new connections. child will continue to accept TIME requests from other clients, since it is a concurrent server. The main (parent) process now handles the connected client.
9. After clearing the buffer memory area using memset() function, TIME request is received from client using recvfrom (s, buffer, 1024, 0, (struct sockaddr*) & cliaddr, &t.
10. prints the formatted string TIME to buffer.
11. Sends back UPDATED CURRENT TIME to client using sendto (s, buffer, sizeof(buffer), 0, (struct sockaddr*) & cliaddr, sizeof(cliaddr)
12. close the socket using close(int sockfd) function.

Teacher's Signature

Date _____

Expt. No. _____

Page No. 11.

Result.

Implement concurrent time server using UDP as transport layer protocol by creating program at remote server.

Teacher's Signature _____

Date

Page No.

Expt. No. 6

Multiuser chat server and client

Aim

To implement a multi user chat seaver using TCP as transport layer protocol.

problem description:- Using TCP socket, create connection between multiple clients and single server.

Algorithm - TCP server

1. Create a socket for TCP using the function call, socket(AFINET, SOCKSTREAM, 0);
2. The memset() function fills the first n bytes of memory area pointed to by addr with constant byte 0.
3. Initialize the structure sockaddr_in member of sinfamily, sin_addr, sin_port.
4. Bind the socket to its port using bind(int sockfd, (struct sockaddr*) &ser_addr, sizeof(ser_addr));
5. Listen for any active client connections using listen(int sockfd, int backlog); backlog argument defines the maximum length to which queue of pending connections for sockfd may grow.
6. Server infinitely accepts client connections using accept function call as follows accept(int sockfd, (struct sockaddr*) &cl_addr, &sizeof(cl
7. After accepting client connections, inet_ntop() function is used to convert clients network address structure, src in the af address family into a character string. The resulting string is copied to the buffer pointed to by dst, which must be a non-null pointer. The caller specifies the number of bytes available in this buffer in argument size.

Teacher's Signature