SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NETWORKS LAB
EXERCISE 2

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

Aim:

To learn and understand the use of system calls used in computer networks.

1. **Socket()**

i. Description: used to create an endpoint for communication

ii. Header files: sys/socket.h

iii. Syntax: socket(args)

1. Parameters: int domain, int protocol

iv. Explanation of parameters:

1. Domain: used to specify the communication domain
2. Protocol: this is automatically selected by the domain parameter

v. Return value: returns a descriptor

vi. Structures used if any: None

2. Bind()

i. Description: used to bind a name/address as specified to the socket

ii. Header files: sys/socket.h

iii. Syntax: bind(args)

1. Parameters: int sockfd, struct sockaddr *addr, socklen_t addrlen

iv. Explanation of parameters:

1. Sockfd: this is the file descriptor of the socket
2. Addr: this is the address to be binded for the socket
3. Addrlen: the size of the struct addr

v. Return value: 0 on success, -1 on error.

vi. Structures used if any: struct sockaddr

3. Listen()

i. Description: used to listen for connections on a socket

ii. Header files: sys/socket.h

iii. Syntax:

1. Parameters: int sockfd, int backlog

iv. Explanation of parameters:

1. Sockfd: this is the file descriptor of the socket
2. Backlog: The backlog argument defines the maximum length to which the queue of pending connections for sockfd may grow.

v. Return value: 0 on success, -1 on failure

vi. Structures used if any: None

4. Connect()

i. Description: used to initiate a connection on a socket referred by the file descriptor of the socket

ii. Header files: sys/socket.h

iii. Syntax:

1. Parameters: int sockfd, struct sockaddr *addr, socklen_t addrlen

iv. Explanation of parameters:

1. Sockfd: file descriptor of the socket
2. Addr: this is the address which is binded to the socket
3. Addrlen: the size of the struct addr

v. Return value: 0 if success, -1 if failure

vi. Structures used if any: struct sockaddr

5. Accept()

i. Description: accepts a connection on a socket, and extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket.

ii. Header files: sys/socket.h

iii. Syntax: accept(args)

1. Parameters: int sockfd, struct sockaddr *restrict addr, socklen *restrict addrelen

iv. Explanation of parameters:

1. Sockfd: file descriptor of the socket
2. Addr: this is the address which is binded to the socket
3. Addrlen: the size of the struct addr

v. Return value: returns the file descriptor of the accepted socket, -1 on failure.

vi. Structures used if any: struct sockaddr

6. Close()

i. Description: used to shut down the socket

ii. Header files: unistd.h

iii. Syntax: close(args)

1. Parameters: int socket

iv. Explanation of parameters:

1. Socket: file descriptor of the sockets

v. Return value: 0 if success, -1 if failure.

vi. Structures used if any: None

7. Bzero()

i. Description: The bzero() function erases the data in the n bytes of the memory starting at the location pointed to by s, by writing zeros (bytes containing '\0') to that area.

ii. Header files: strings.h

iii. Syntax: bzero(args)

1. Parameters: void *s, size_t n

iv. Explanation of parameters:

1. s: location of the string

2. n: size of the string in bytes

v. Return value: void

vi. Structures used if any: None

8. htons, htonl, ntohs, ntohl()

i. Description: These functions shall convert 16-bit and 32-bit quantities between network byte order and host byte order.

ii. Header files: arpa/inet.h

iii. Syntax: uint32_t htonl(uint32_t hostlong); uint16_t htons(uint16_t hostshort); uint32_t ntohl(uint32_t netlong); uint16_t ntohs(uint16_t netshort);

1. Parameters: uint32_t, uint16_t

iv. Explanation of parameters:

1. Any integer that needs to be converted.

v. Return value: the converted integer

vi. Structures used if any: None

9. Read()

i. Description: used to read n bytes from the socket to the buffer specified.

ii. Header files: unistd.h

iii. Syntax: read(args)

1. Parameters: int fd, void *buf, size_t count

iv. Explanation of parameters:

1. fd: file descriptor of the socket
2. buf: the buffer into which the read items are to be stored
3. count: the size to be read

v. Return value: size read from the socket

vi. Structures used if any: None

10. Write()

i. Description: used to write n bytes to the socket to the buffer specified.

ii. Header files: unistd.h

iii. Syntax: write(args)

1. Parameters: int fd, void *buf, size_t count

iv. Explanation of parameters:

1. fd: file descriptor of the socket
2. buf: the buffer into which the read items are to be stored
3. count: the size to be read

v. Return value: size written to the socket

vi. Structures used if any: None

11. Send()

i. Description: send

ii. Header files: sys/socket.h

iii. Syntax: send(args)

1. Parameters: sockfd, buf, len, flags

iv. Explanation of parameters:

1. Sockfd: file descriptor of the socket
2. Buf: the message to be sent is stored here
3. Len: length of the buffer

. Flags: flags to be used

v. Return value: returns the number of bytes sent, -1 on failure

vi. Structures used if any: None

12. Receive()

i. Description: Used to receive a message from the socket

ii. Header files: sys/socket.h

iii. Syntax:

1. Parameters: sockfd, buf, len, flags

iv. Explanation of parameters:

1. Sockfd: file descriptor of the socket
2. Buf: the message to be sent is stored here
3. Len: length of the buffer
4. Flags: flags to be used

v. Return value: Returns the number of bytes received, or -1 if an error occured

vi. Structures used if any: None

13. Sendto()

i. Description: The send() call is used only when the socket is in connected state.

ii. Header files: sys/socket.h

iii. Syntax:

1. Parameters: sockfd, buf, len, flags

iv. Explanation of parameters:

1. Sockfd: file descriptor of the socket
2. Buf: the message to be sent is stored here
3. Len: length of the buffer
4. Flags: flags to be used

v. Return value: On success returns the number of bytes sent, else will return -1

vi. Structures used if any: None

14. Receivefrom()

i. Description: used to receive messages from a socket, and may be used to receive data on a socket whether or not it is connection-oriented.

ii. Header files: sys/socket.h and sys/types.h

iii. Syntax:

1. Parameters: sockfd, buf, len, flags

iv. Explanation of parameters:

1. Sockfd: file descriptor of the socket
2. Buf: the message to be sent is stored here
3. Len: length of the buffer
4. Flags: flags to be used

v. Return value: Returns number of bytes received or will simply return -1 if an error occurs

vi. Structures used if any: None

15. Select()

i. Description: Select command allows to monitor multiple file descriptors, waiting until one of the file descriptors become active.

ii. Header files: sys/select.h

iii. Syntax:

1. Parameters: nfds, readfds, writefds, exceptfds

iv. Explanation of parameters:

1. Nfds: Used to set the highest numbered file descriptor

In any of the three sets, plus 1

2. Readfds: Used to check if the ready for reading

3. Writedfs: The file descriptors in this set are watched to see if they are ready for writing

4. Exceptfds: The file descriptors in this set are watched for exceptional conditions

v. Return value: Returns the number of file descriptors contained in the three returned descriptors sets. On error returns -1

vi. Structures used if any: None

16. Setsockopt()

i. Description: This is used to manipulate the options associated with a socket

ii. Header files: sys/types.h and sys/socket.h

iii. Syntax:

1. Parameters: socket, level, optname, optval, optlen

iv. Explanation of parameters:

1. Socket: stores the filedescriptor

2. Level: specifies the protocol level at which the option resides

3. Optname: specifies the a single option to set

4. Optval: stores the value argument for the socket

5. Optlen: defines length

v. Return value: Upon successful completion it will return 0, else -1 will be returned and errno set to indicate the error

vi. Structures used if any: None

17. Fcntl()

i. Description: Used to perform file descriptor manipulations

ii. Header files: fcntl.h

iii. Syntax:

1. Parameters: fd, cmd

iv. Explanation of parameters:

1. Fd: holds the value for file descriptor
2. Cmd: Decides the functionality of the entire function

v. Return value: Will depend on the cmd passed during the beginning, Other wise -1 will be returned in case of any error

vi. Structures used if any: None

18. getpeername()

i. Description: Is used to return the address of the peer connected to the socket sockfd, in the buffer pointed to by addr

ii. Header files: sys/socket.h

iii. Syntax:

1. Parameters: sockfd, addr, addrlen

iv. Explanation of parameters:

1. Sockfd: Holds the socket number who's address is to be returned
2. Addr: refers to the amount of space

3. Addrlen: Initialized to indicate the amount of space pointed to by addr

v. Return value: On success, zero is returned. On error, -1 is returned and errno is set to the indicate the error

vi. Structures used if any: None

19. gethostname()

i. Description: These are used to access the system hostname

ii. Header files: unistd.h

iii. Syntax:

1. Parameters: name, len

iv. Explanation of parameters:

1. Name: Used to pass the hostname
2. Len: specifies the length of the byte

v. Return value: On success zero is returned. On error -1 is returned

vi. Structures used if any: None

20. gethostbyname()

i. Description: It is used to return a structure of type hostent for the given host name

ii. Header files: netbd.h

iii. Syntax:

1. Parameters: name

iv. Explanation of parameters:

1. Name: It is used to specify the name of host or Itv4 address

v. Return value: Returns the hostent structure or a numm pointer if an error occurs

vi. Structures used if any: hostent

21. gethostbyaddr()

i. Description: Returns a structure of type hostent for the given host address addr of length len and address type type

ii. Header files: netdb.h

iii. Syntax:

1. Parameters: addr, len, type

iv. Explanation of parameters:

1. Addr: specifies the address we are looking for
2. Len: length of the address
3. Type: type of the address

v. Return value: Returns the required structure with the given address, else return NULL if not found

vi. Structures used if any: hostent

Result:

Learned about the use of system calls used in computer networks

Learning outcome:

Learnt the various network related commands
Learnt the various options associated with the network related commands
Learnt and implemented the network commands on the terminal.