Computer Networks - II LAB PROGRAMS (VI Semester)

1. File transfer using PIPES

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#define maxsize 1000
char buffer[1000];
void client( int readfd, int writefd )
{
        printf("\nGive pathname: ");
        fflush(stdout);
        fgets(buffer, sizeof(buffer), stdin);
        printf("\nServer Online\nProcessing request...\n\n");
        write(writefd, buffer, sizeof( buffer ) );
        while( read( readfd, buffer, sizeof( buffer ) ) > 0 )
                printf( "%s", buffer);
}
void server( int readfd, int writefd )
{
        FILE * fp;
        char line[1000];
        read( readfd, buffer, sizeof( buffer ) );
        if( strchr( buffer, '\n' ) )
                *strchr( buffer, '\n' ) = 0;
        fp = fopen( buffer, "r" );
        if( fp == NULL )
        {
                strcpy( buffer, "Cannot open file" );
                write( writefd, buffer, strlen( buffer ) );
                exit(1);
        }
        else
        {
                while( fgets( line, sizeof( line ), fp ) != NULL )
                        write( writefd, line, sizeof( line ) );
                printf("SERVER: Transfer completed\n");
        }
        printf("\n");
}

int main()
{
        int pipe1[2], pipe2[2];
        int childpid;
        int status;
        pipe( pipe1 );
        pipe( pipe2 );
        printf("\nClient Online\n");
        childpid = fork();
        if( childpid > 0 )
        {
                close( pipe1[0] );
                close( pipe2[1] );
                client( pipe2[0], pipe1[1] );
                wait( &status );
                exit(0);
        }
        else
        {
                close( pipe1[1] );
                close( pipe2[0] );
                server( pipe1[0], pipe2[1] );
                exit(0);
        }
}
```

----

2. File transfer b/w Client and Server using FIFO

```
---------------CLIENT
/*Client*/
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>
#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666
char fname[256];
int main()
{
  ssize_t n;
  char buff[512];
  int readfd,writefd;
  printf("Trying to Connect to Server..\n");
  writefd = open(FIFO1, O_WRONLY, 0);
  readfd  = open(FIFO2, O_RDONLY, 0);
  printf("Connected..\n");
  printf("Enter the filename to request from server: ");
  scanf("%s",fname);
  write(writefd, fname, strlen(fname));
  printf("Waiting for Server to reply..\n");
  while((n=read(readfd,buff,512))>0)
    write(1,buff,n);
  close(readfd);
  close(writefd);
  return 0;
}

-----------------SERVER
/*Server*/
#include<stdio.h>
#include<unistd.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<string.h>
#define FIFO1 "fifo1"
#define FIFO2 "fifo2"
#define PERMS 0666
char fname[256];
int main() {
  int readfd, writefd, fd;
  ssize_t n;
  char buff[512];
  if (mkfifo(FIFO1, PERMS)<0)
    printf("Cant Create FIFO Files\n");
  if (mkfifo(FIFO2, PERMS)<0)
    printf("Cant Create FIFO Files\n");
  printf("Waiting for connection Request..\n");
  readfd =open(FIFO1, O_RDONLY, 0);
  writefd=open(FIFO2, O_WRONLY, 0);
  printf("Connection Established..\n");
  read(readfd, fname, 255);
  printf("Client has requested file %s\n", fname);
  if ((fd=open(fname,O_RDWR))<0) {
    strcpy(buff,"File does not exist..\n");
    write(writefd, buff, strlen(buff));
  } else {
    while((n=read(fd, buff,512))>0)
      write(writefd, buff, n);
  }
  close(readfd);  unlink(FIFO1);
  close(writefd);  unlink(FIFO2);
}
```

----

3. File transfer b/w Client and Server using MESSAGE QUEUE

```
-------------------CLIENT
//Client using Message Queue
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include <string.h>
#define MSGSZ    10000
// Declare the message structure.
typedef struct msgbuf {
        long    mtype;
        char    mtext[MSGSZ];
        } message_buf;
int main()
{
    int msqid1,msqid2;
    int msgflg = IPC_CREAT | 0666;
    key_t key1,key2;
    message_buf buf;
    size_t buf_length;
  /*Key1 for MQ1 & Key2 for MQ2*/
    key1 = 1234;
    key2=5678;
    if ((msqid1 = msgget(key1, msgflg )) < 0) {
        printf("CLIENT: Can't open output message queue \n");
        return 0;
    }
    else
     printf("CLIENT : Output message queue opened successfully\n");
    if ((msqid2= msgget(key2, msgflg )) < 0) {
        printf("CLIENT : Can't open input message queue \n");
        return 0;
    }
    else
     printf("CLIENT : Input message queue opened successfully\n");
        /*send message type 1*/
        buf.mtype = 1;
    printf("\nEnter the filename : ");
    scanf("%s",buf.mtext);
    buf_length = strlen(buf.mtext) + 1 ;
    /* Send  the filename*/
    if (msgsnd(msqid1, &buf, buf_length, IPC_NOWAIT) < 0) {

        printf("CLIENT : Error on sending filename\n");
        return 0;
     }
   else
      printf("CLIENT :Filename sent to the server.. waiting for reply..\n");
       if (msgrcv(msqid2, &buf, MSGSZ, 2, 0) < 0) {
        printf("CLIENT : Error on receiving the reply..\n");
        return 0;
     }
  else
  {
    printf("CLIENT : Reply from server:\n");
    fputs(buf.mtext,stdout);
    printf("\n\n");
   }
     return 0;
}

--------------------SERVER
//Server using Message Queue
#include <sys/types.h>
#include<string.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <stdio.h>
#include<fcntl.h>
#define MSGSZ     10000
/* Declare the message structure.*/
typedef struct msgbuf {
    long    mtype;
    char    mtext[MSGSZ];
} message_buf;
int main()
{
    int msqid1,msqid2,f1,filesize,n;
    key_t key1,key2;
    message_buf  buf;
    size_t buf_length;
    /*Key1 for MQ1 & Key2 for MQ2*/
    key1 = 1234;
    key2 = 5678;
    if ((msqid1 = msgget(key1, 0666)) < 0) {
         printf("SERVER : Can't open input message queue \n");
        return 0;
    }
    else
      printf("SERVER : Input message queue opened successfully\nSERVER : Waiting for client 
request..\n");
```

```
    /*Receive an answer of message type 1.*/
    if (msgrcv(msqid1, &buf, MSGSZ, 1, 0) < 0) {
        printf("SERVER : Can't receive the message..\n");
        return 0;
    }
   if ((msqid2= msgget(key2, 0666 )) < 0) {
        printf("SERVER : Can't open output message queue \n");
        return 0;
    }
    else
     printf("SERVER : Output message queue opened succesfully\n");
     buf.mtype = 2;
      if((f1=open(buf.mtext,O_RDONLY))!=-1)
             {
        printf("\nSERVER : %s is found \nTransfering the contents.. \n",buf.mtext);
                          filesize=lseek(f1,0,2);
                          printf("\nSERVER : File size is %d\n",filesize);
                          lseek(f1,0,0);//rewind file pointer to beginning
                          n=read(f1,buf.mtext,filesize);
               buf_length = strlen(buf.mtext) + 1 ;
               if (msgsnd(msqid2, &buf, buf_length, IPC_NOWAIT) < 0) {
                printf("SERVER : Error on message sending..\n");
                 return 0;
                }
              else
               printf("SERVER : File contents transfered successfully..\n\n");
          }
        else
       {
                  printf("SERVER : File %s not found\n",buf.mtext);
             strcpy(buf.mtext, "File Not Found");
             buf_length = strlen(buf.mtext) + 1 ;
             if (msgsnd(msqid2, &buf, buf_length, IPC_NOWAIT) < 0) {

             printf("SERVER : Error on message sending.. \n");
               return 0;
              }
            else
             printf("SERVER : Reply sent to client successfully..\n\n");
       }
       return 0;
}
```

----

4. File transfer b/w Client and Server using SOCKETS

----------------CLIENT

```
/* CLIENT */
#include<sys/socket.h>
#include<sys/types.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
int main(int argc,char *argv[])
{
int create_socket,cont;
int bufsize = 1024;
char *buffer = malloc(bufsize);
char fname[256];
struct sockaddr_in address;
if ((create_socket = socket(AF_INET,SOCK_STREAM,0)) > 0)
printf("The Socket was created\n");
address.sin_family = AF_INET;
address.sin_port = htons(11000);
inet_pton(AF_INET,argv[1],&address.sin_addr);
if (connect(create_socket,(struct sockaddr *) &address, sizeof(address)) == 0)
printf("The connection was accepted with the server %s...\n",argv[1]);
printf("Enter The Filename to Request : "); scanf("%s",fname);
send(create_socket, fname, sizeof(fname), 0);
printf("Request Accepted... Receiving File...\n\n");
printf("The contents of file are...\n\n");
while((cont=recv(create_socket, buffer, bufsize, 0))>0) {
write(1, buffer, cont);
}
printf("\nEOF\n");
return close(create_socket);
}

--------------------SERVER
/* SERVER */
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<sys/stat.h>
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<fcntl.h>

int
main ()
{
  int cont, create_socket, new_socket, addrlen, fd;
```

```
 int bufsize = 1024;
 char *buffer = malloc (bufsize);
 char fname[256];
 struct sockaddr_in address;
 if ((create_socket = socket (AF_INET, SOCK_STREAM, 0)) > 0)
   printf ("The socket was created\n");
 address.sin_family = AF_INET;
 address.sin_addr.s_addr = INADDR_ANY;
 address.sin_port = htons (11000);
 if (bind (create_socket, (struct sockaddr *) &address, sizeof (address)) == 0)
   printf ("Binding Socket\n");
 listen (create_socket, 3);
 addrlen = sizeof (struct sockaddr_in);
 new_socket = accept (create_socket, (struct sockaddr *) &address, &addrlen);
 if (new_socket > 0)
   printf ("The Client %s is Connected...\n", inet_ntoa (address.sin_addr));
 recv (new_socket, fname, 255, 0);
 printf ("A request for filename %s Received..\n", fname);
 if ((fd = open (fname, O_RDONLY)) < 0)
   {
     perror ("File Open Failed");
     exit (0);
   }
 while ((cont = read (fd, buffer, bufsize)) > 0)
   {
     send (new_socket, buffer, cont, 0);
   }
 printf ("Request Completed\n");
 close (new_socket);
 return close (create_socket);
}
```

----

5. DISTANCE Vector Routing

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int dm[20][20], no, i, j, k, source, dest;
void dvr(int , int);

struct node
{
 int dist[20];
 int from[20];
} route[10];
```

```
int main ()
{
  cout << "Enter the number of nodes:\t";
  cin >> no;
  cout << "Enter the Source and Destination nodes:\t";
  cin >> source >> dest;
  cout << "\nEnter the distance matrix:\n";
  for (i = 0; i < no; i++)
    {
      for (j = 0; j < no; j++)
	{
	  cin >> dm[i][j];
	  dm[i][i] = 0;
	  route[i].dist[j] = dm[i][j];
	  route[i].from[j] = j;
	}
    }
  int flag;
  do
    {
      flag = 0;
      for (i = 0; i < no; i++)
	{
	  for (j = 0; j < no; j++)
	    {
	      for (k = 0; k < no; k++)
		{
		  if ((route[i].dist[j]) >
		      (route[i].dist[k] + route[k].dist[j]))
		    {
		      route[i].dist[j] =
			(route[i].dist[k] + route[k].dist[j]);
		      route[i].from[j] = k;
		      flag = 1;
		    }
		}
	    }
	}
    }
  while (flag);
  for (i = 0; i < no; i++)
    {
      cout << "Router Info for Router: " << i + 1 << endl;
      cout << "Dest\t NextHop\t Dist" << endl;
      for (j = 0; j < no; j++)
	cout << "  " << j + 1 << " \t    " << route[i].from[j] +
               1 << " \t \t   " << route[i].dist[j] << endl;
    }
cout << "The shortest path from Source to Destination is:\n";
cout  << source;
dvr(source, dest);
/*
  for (i = 0; i < no; i++)
    {
      if (source-1 == i)
             {
             for(j=0;j<no;j++)
             {
               //cout << route[i].from[j]+1 << "\t" << route[i].dist[j] << endl;
               if (j+1 == dest)
                { ns = route[i].from[j]+1;
                  cost = route[i].dist[j];
                  cout << "-->" << ns;
                }
             }
             }
    } */
return 0;
}

void
dvr (int source, int dest)
{
  int cost = 0, ns = 0, ns1 = 0;
  for (i = 0; i < no; i++)
    {
      if (source - 1 == i)
             {
               for (j = 0; j < no; j++)
                 {
                  if (j + 1 == dest)
                         {
                 //if (route[i].from[j] + 1 == dest)
                 //{ exit(0);}
                          ns = route[i].from[j] + 1;
                          //cost = route[i].dist[j];
                          cout << "-->" << ns;
                 if (ns == dest)
                          { cout<<endl;exit(0);}
                          dvr (ns , dest);
                         }
                 }
             }
  }
}
```

----

6. LINK Vector Routing

```
#include<stdio.h>
int i=0,j=0,k=0;
int
main ()
{
 // freopen("input.txt","r",stdin);
 int n, a[10][10], i, j, d[10], p[10], s[10];
 printf ("\n ENTER THE NO.OF NODES: ");
 scanf ("%d", &n);
 printf ("\n ENTER THE MATRIX ELEMENTS: ");
 for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
        scanf ("%d", &a[i][j]);
  }
 printf ("\n");
 for (i = 0; i < n; i++)
  {
    printf ("The Link State Packets for Router %d\n", i + 1);
    printf ("NODES | DISTANCE\n");
    printf ("--------------\n");
    for (j = 0; j < n; j++)
        {
          if (a[i][j] != 0 && a[i][j] != 9999)
           {
             printf ("%d     |     %d\n", j + 1, a[i][j]);
           }
        }
    printf ("--------------\n");
  }
 for (i = 0; i < n; i++)
  {
    for (j = 0; j < n; j++)
        {
          for (k = 0; k < n; k++)
           {
             if (a[i][j] > a[i][k] + a[k][j])
                 a[i][j] = a[i][k] + a[k][j];
           }
        }
  }
 printf ("\nShortest Distance for \n");
 for (i = 0; i < n; i++)
  {
    printf ("Router %d\n", i + 1);
    printf ("--------------\n");
    printf ("NODES | Shortest dist.\n");
    for (j = 0; j < n; j++)
        {
          if (a[i][j]!=0) { printf ("%d     |     %d\n", j + 1, a[i][j]); }
        }
    printf ("--------------\n");
  }
 return 0;
}
```

----

7. CRC Error Detection (CCITT-16)

```
#include<iostream>
#include<string.h>
using namespace std;
int
crc (char* ip, char* op, char* poly, int mode)
{
  strcpy (op, ip);
  if (mode)
    {
      for (int i = 1; i < strlen (poly); i++)
        strcat (op, "0");
    }
  for (int i = 0; i < strlen (ip); i++)
    {
      if (op[i] == '1')
        {
          for (int j = 0; j < strlen (poly); j++)
            {
              if (op[i + j] == poly[j])
                op[i + j] = '0';
              else
                op[i + j] = '1';
            }
        }
    }
  for (int i = 0; i < strlen (op); i++)
    if (op[i] == '1')
      return 0;
    //else
    return 1;
}

int
main ()
{
  char ip[50], op[50], recv[50];
  char poly[] = "10001000000100001";
  cout << "Enter message: ";
  cin >> ip;
  crc (ip, op, poly, 1);
  cout << "Transmitted message is:  " << ip << op + strlen (ip) << endl;
  cout << "\nEnter recieved message in binary:  " << endl;
  cin >> recv;
  if (crc (recv, op, poly, 0))
    cout << "--No error in Transmission--\n";
  else
    cout << "--Error in transmission--\n";
return 0;
}
```

----

8. Internet Check-Sum

```
#include<stdio.h>
#include<iostream>
using namespace std;

unsigned short check()
{
int sum = 0;
unsigned short int fields[10];

for(int i=0;i<9;i++)
{
cout << "Fields:\t" << i+1 << endl;
scanf("%x",&fields[i]);
sum += (unsigned short)fields[i];
while(sum>>16)
 sum = (sum & 0xFFFF) + (sum >> 16);
}
sum = ~sum;
cout << "\nThe Checksum is :\t" << sum << endl;
return (unsigned short) sum;
}

int main()
```

```
{
unsigned short res1, res2;
cout << "Sender:" << endl;
res1 = check();
cout << "Reciever:" << endl;
res2 = check();
if(res1 == res2)
 cout << "NO ERROR\n";
else
 cout << "ERROR\n";
}
```

----

9. Hamming Code

```
#include<stdio.h>
#include<stdlib.h>
main ()
{
  int i, a[4], b[4], r[4], s[3];
  printf ("\nEnter 4 bit data word:");
  for (i = 3; i >= 0; i--)
    scanf ("%d", &a[i]);
  r[0] = (a[2] + a[1] + a[0]) % 2;
  r[1] = (a[2] + a[1] + a[3]) % 2;
  r[2] = (a[0] + a[1] + a[3]) % 2;
  printf ("\n7 bit hamming codeword is:\n");
  for (i = 3; i >= 0; i--)
    printf ("%d\t", a[i]);
  for (i = 2; i >= 0; i--)
    printf ("%d\t", r[i]);
  printf ("\n");
  printf ("\nEnter 4 bit recieved word:");
  for (i = 3; i >= 0; i--)
    scanf ("%d", &b[i]);
  s[0] = (b[2] + b[1] + b[0] + r[0]) % 2;
  s[1] = (b[3] + b[2] + b[1] + r[1]) % 2;
  s[2] = (b[0] + b[1] + b[3] + r[2]) % 2;
  printf ("\nSyndrome is:\n");
  for (i = 2; i >= 0; i--)
    printf ("%d\t", s[i]);
  if ((s[2] == 0) && (s[1] == 0) && (s[0] == 0))
    printf ("\nRecieved data is error free\n");
  else
   {
     if ((s[2] == 1) && (s[1] == 0) && (s[0] == 1))
          {
            if (b[0] == 1)
              b[0] = 0;
            else
              b[0] = 1;
            printf
              ("\nRecieved word has error in 1st bit(b0) position from right\n");
          }
      if ((s[2] == 1) && (s[1] == 1) && (s[0] == 1))
          {
            if (b[1] == 1)
              b[1] = 0;
            else
              b[1] = 1;
            printf
              ("\nRecieved word has error in 2nd bit(b1) position from right\n");
          }
      if ((s[2] == 0) && (s[1] == 1) && (s[0] == 1))
          {
            if (b[2] == 1)
              b[2] = 0;
            else
              b[2] = 1;
            printf
              ("\nRecieved word has error in 3rd bit(b2) position from right\n");
          }
      if ((s[2] == 1) && (s[1] == 1) && (s[0] == 0))
          {
            if (b[3] == 1)
              b[3] = 0;
            else
              b[3] = 1;
            printf
              ("\nRecieved word has error in 4th bit(b3) position from right\n");
          }
      printf ("Corrected recieved word is:\n");
      for (i = 3; i >= 0; i--)
          printf ("%d\t", b[i]);
    }
}
```

----

10. Leaky Bucket

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<unistd.h>
```

```
#define bucketSize 512
void bktInput(int a,int b)
{
        if (a > bucketSize)
                std::cout << "\n\t\tBucket overflow\n";
        else
        {
        sleep(1);
        while (a > b)
                {
                        std::cout << "\n\t\t" << b << " bytes outputted.";
                        a -= b;
                        sleep(1);
                }
                if (a > 0)
                        std::cout << "\n\t\tLast " << a << " bytes sent\t";
                std::cout << "\n\t\tBucket output successful\n\n";
        }
}
int main()
{
        int                op      , pktSize;
        std::cout << "Enter output rate : ";
        std::cin >> op;
        for (int i = 1; i <= 4; i++) {
                sleep(1);
                pktSize=rand()%100;
                std::cout << "\nPacket no " << i << "\tPacket size = " << pktSize;
                bktInput(pktSize, op);
        }
        return 0;
}
```

----

11. Multicast Routing

```
------------------LISTENER
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include<stdlib.h>

#define HELLO_PORT 6000
```

```
#define HELLO_GROUP "225.0.0.38"
#define MSGBUFSIZE 256

main(int argc, char *argv[])
{
    struct sockaddr_in addr;
    int fd, nbytes,addrlen;
    struct ip_mreq mreq;
    char msgbuf[MSGBUFSIZE];

    u_int yes=1;            /*** MODIFICATION TO ORIGINAL */

    /* create what looks like an ordinary UDP socket */
    if ((fd=socket(AF_INET,SOCK_DGRAM,0)) < 0) {
            perror("socket");
            exit(1);
    }


/**** MODIFICATION TO ORIGINAL */
   /* allow multiple sockets to use the same PORT number */
   if (setsockopt(fd,SOL_SOCKET,SO_REUSEADDR,&yes,sizeof(yes)) < 0) {
      perror("Reusing ADDR failed");
      exit(1);
      }
/*** END OF MODIFICATION TO ORIGINAL */

    /* set up destination address */
    memset(&addr,0,sizeof(addr));
    addr.sin_family=AF_INET;
    addr.sin_addr.s_addr=htonl(INADDR_ANY); /* N.B.: differs from sender */
    addr.sin_port=htons(HELLO_PORT);

    /* bind to receive address */
    if (bind(fd,(struct sockaddr *) &addr,sizeof(addr)) < 0) {
            perror("bind");
            exit(1);
    }

    /* use setsockopt() to request that the kernel join a multicast group */
    mreq.imr_multiaddr.s_addr=inet_addr(HELLO_GROUP);
    mreq.imr_interface.s_addr=htonl(INADDR_ANY);
    if (setsockopt(fd,IPPROTO_IP,IP_ADD_MEMBERSHIP,&mreq,sizeof(mreq)) < 0) {
            perror("setsockopt");
            exit(1);
    }
    /* now just enter a read-print loop */
    while (1) {
            addrlen=sizeof(addr);
            if ((nbytes=recvfrom(fd,msgbuf,MSGBUFSIZE,0,(struct sockaddr *)
&addr,&addrlen)) < 0) {
                perror("recvfrom");
                exit(1);
            }
            puts(msgbuf);
    }
}

--------------------SENDER
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <time.h>
#include <string.h>
#include <stdio.h>
#include<string.h>
#include<stdlib.h>

#define HELLO_PORT 6000
#define HELLO_GROUP "225.0.0.38"

main(int argc, char *argv[])
{
    struct sockaddr_in addr;
    int fd, cnt;
    struct ip_mreq mreq;
    char *message="Hello, World!";

    /* create what looks like an ordinary UDP socket */
    if ((fd=socket(AF_INET,SOCK_DGRAM,0)) < 0) {
            perror("socket");
            exit(1);
    }

    /* set up destination address */
    memset(&addr,0,sizeof(addr));
    addr.sin_family=AF_INET;
    addr.sin_addr.s_addr=inet_addr(HELLO_GROUP);
    addr.sin_port=htons(HELLO_PORT);

    /* now just sendto() our destination! */
    while (1) {
```

```
if (sendto(fd,message,strlen(message),0,(struct sockaddr *) &addr,sizeof(addr)) < 0)
                    {
                perror("sendto");
                exit(1);
            }
            sleep(1);
    }
}
```

Compiled by-
N.Chandra Kanth