197103_Abhishek	197249 Mahitha Maram
197106 Aditya Joshi	197249 Mahitha Maram file path only
197108 Akshat Dhiman	197255 Pravalika Narala
197111 Anubrata Seal	197256 Pavan
197112 Archana Banoth	197257 Nemi Brahmachari
197114_samba siva reddy asam	197259 Manoj Kumar Reddy
197119 sreyas bekkam	197260 Rithin
197122_chandrapal_baghel	197261 Prayag Patel
197125 Deekshita Tirumala	197266_ShreeRam Mohanty
197134 Shantanu	197269 Ritik Kumar
197138 Ishan Joshi	197273 Saswat Das
197139 Sai Vivek	197276 Kulshreshtha
197149 MAMIDI SUSHITH REDDY	197278 Sowmya
197150 Meda Jaithra	197280 kishore surapally
197152 Chirantan	197281 waseem syed
197153 N Santhosh Kumar	197285 Vandana
197154 Rahul	Adith 197104
197156 Nayan Narzary	Aditya singh
197160 Gnana Samhitha	Akash 197245
197161 Thanmayee	anil kumar aska
197164_Pushpraj Bhuriya	Ankit Reddy
197167 RAJA REDDY PUNDRA	Arpit Bohra
197171 Sahukari SaiVamsi	Ayush Agrawal
197179 kushi varshith	Chinmay Hardikar 197131
197185 Shiva Shankar Vaddepallly	D Saiteja 197123
197203	Dasari suma
197205 Aditya Srivastav	Lokesh Tejavath
197207 Akhil Vardhan Mallipeddu	meghana 197159
197207 Akhil Vardhan Mallipeddu	Naman Balai
197210 Ankit Gond	Priyanshu Khetan
197211 Apurv Jain	Racha Vinay 197264
197212 Ramakrishna	Rahul Kolluru
197213 Aryan Karki	Ravirala Bhargavi
197214 Ashiqa	S.S.Lakshayapriya 197270
197215_Ashutosh Chandra	Sai Bharath Reddy 197247
197216 Ayush Singhal	Sai Ganesh Kasina
197217 Saipreetham Bachu	Samarth Garg
197220 Suchith Reddy	Sangam Kushwaha
197221 chandana	Sanjana Kosuru
197223 Darshan Solanki	Santosh Kumar 197118
197226 Devansh Ahuja	shivaji 197275
197230 Chaitanya Hardikar	shweta thote
197234 Saikalyan Induri	Soumyadip Payra
197235 Ishita Gupta	Sudireddy Dinesh Reddy 197279
197238 Joy Chhajed	Vedant Gandhi
197242 Keshav Ganesh	Venkat Sai Naik Tejavath
197243 Rahul Khatav	Vishwas Gajawada
197248 Venkatasai Maddisetty	yashpal

UNIX SOCKET CONNECTION ORIENTED SERVER (usage -: "./a.out")

```
-------------------------------------------------------------------------
```

```
      #define ADDRESS  "mysocket"

      int  usfd;
      struct sockaddr_un userv_addr,ucli_addr;
      int userv_len,ucli_len;

      usfd = socket(AF_UNIX , SOCK_STREAM , 0);
      perror("socket");

      bzero(&userv_addr,sizeof(userv_addr));

      userv_addr.sun_family = AF_UNIX;
      strcpy(userv_addr.sun_path, ADDRESS);
      unlink(ADDRESS);
      userv_len = sizeof(userv_addr);

      if(bind(usfd, (struct sockaddr *)&userv_addr, userv_len)==-1)
      perror("server: bind");

      listen(usfd, 5);

      ucli_len=sizeof(ucli_addr);

      int nusfd;
      nusfd=accept(usfd, (struct sockaddr *)&ucli_addr, &ucli_len);
```

UNIX SOCKET CONNECTION ORIENTED CLIENT (usage -: "./a.out")

```
-----------------------------------------------------------------
      #define ADDRESS     "mysocket"

      int usfd;
      struct sockaddr_un userv_addr;
      int userv_len,ucli_len;

      usfd = socket(AF_UNIX, SOCK_STREAM, 0);

      if(usfd==-1)
      perror("\nsocket  ");

      bzero(&userv_addr,sizeof(userv_addr));
      userv_addr.sun_family = AF_UNIX;
      strcpy(userv_addr.sun_path, ADDRESS);

      userv_len = sizeof(userv_addr);

      if(connect(usfd,(struct sockaddr *)&userv_addr,userv_len)==-1)
      perror("\n connect ");

      else printf("\nconnect succesful");
```

SEND_FD AND RECV_FD

```
-------------------------------------------------------------------------
int send_fd(int socket, int fd_to_send)
 {
  struct msghdr socket_message;
  struct iovec io_vector[1];
  struct cmsghdr *control_message = NULL;
  char message_buffer[1];
  /* storage space needed for an ancillary element with a paylod of
length is CMSG_SPACE(sizeof(length)) */
  char ancillary_element_buffer[CMSG_SPACE(sizeof(int))];
  int available_ancillary_element_buffer_space;

  /* at least one vector of one byte must be sent */
  message_buffer[0] = 'F';
  io_vector[0].iov_base = message_buffer;
  io_vector[0].iov_len = 1;

  /* initialize socket message */
  memset(&socket_message, 0, sizeof(struct msghdr));
  socket_message.msg_iov = io_vector;
  socket_message.msg_iovlen = 1;

  /* provide space for the ancillary data */
  available_ancillary_element_buffer_space = CMSG_SPACE(sizeof(int));
  memset(ancillary_element_buffer, 0,
available_ancillary_element_buffer_space);
  socket_message.msg_control = ancillary_element_buffer;
  socket_message.msg_controllen =
available_ancillary_element_buffer_space;

  /* initialize a single ancillary data element for fd passing */
  control_message = CMSG_FIRSTHDR(&socket_message);
  control_message->cmsg_level = SOL_SOCKET;
  control_message->cmsg_type = SCM_RIGHTS;
  control_message->cmsg_len = CMSG_LEN(sizeof(int));
  *((int *) CMSG_DATA(control_message)) = fd_to_send;

  return sendmsg(socket, &socket_message, 0);
 }




 int recv_fd(int socket)
 {
  int sent_fd, available_ancillary_element_buffer_space;
  struct msghdr socket_message;
  struct iovec io_vector[1];
  struct cmsghdr *control_message = NULL;
  char message_buffer[1];
  char ancillary_element_buffer[CMSG_SPACE(sizeof(int))];

  /* start clean */
  memset(&socket_message, 0, sizeof(struct msghdr));
  memset(ancillary_element_buffer, 0, CMSG_SPACE(sizeof(int)));

  /* setup a place to fill in message contents */
  io_vector[0].iov_base = message_buffer;
  io_vector[0].iov_len = 1;
  socket_message.msg_iov = io_vector;
  socket_message.msg_iovlen = 1;

  /* provide space for the ancillary data */
  socket_message.msg_control = ancillary_element_buffer;
  socket_message.msg_controllen = CMSG_SPACE(sizeof(int));

  if(recvmsg(socket, &socket_message, MSG_CMSG_CLOEXEC) < 0)
   return -1;

  if(message_buffer[0] != 'F')
  {
   /* this did not originate from the above function */
   return -1;
  }

  if((socket_message.msg_flags & MSG_CTRUNC) == MSG_CTRUNC)
  {
   /* we did not provide enough space for the ancillary element array */
   return -1;
  }
  /* iterate ancillary elements */
   for(control_message = CMSG_FIRSTHDR(&socket_message);
       control_message != NULL;
       control_message = CMSG_NXTHDR(&socket_message, control_message))
  {
   if( (control_message->cmsg_level == SOL_SOCKET) &&
       (control_message->cmsg_type == SCM_RIGHTS) )
    {
     sent_fd = *((int *) CMSG_DATA(control_message));
     return sent_fd;
    }
  }

  return -1;
 }
```

UNIX SOCKET CONNECTION LESS SERVER(usage -: "./a.out")

```
      #define ADDRESS  "mysocket"

      int  usfd;
      struct sockaddr_un userv_addr,ucli_addr;
      int userv_len,ucli_len;

      usfd = socket(AF_UNIX , SOCK_DGRAM , 0);
      perror("socket");

      bzero(&userv_addr,sizeof(userv_addr));

      userv_addr.sun_family = AF_UNIX;
      strcpy(userv_addr.sun_path, ADDRESS);
      unlink(ADDRESS);
      userv_len = sizeof(userv_addr);

      if(bind(usfd, (struct sockaddr *)&userv_addr, userv_len)==-1)
      perror("server: bind");

      fgets( buffer , 256 , stdin );
      sendto(usfd , buffer , 256 , 0 , ( struct sockaddr * ) &ucli_addr ,
ucli_len);
      recvfrom(sfd , buffer , 256 , 0 , ( struct sockaddr * ) &ucli_addr
, & uscli_len );
```

UNIX SOCKET CONNECTION LESS CLIENT (usage -:
"./a.out")

```
      #define ADDRESS     "mysocket"

      int usfd;
      struct sockaddr_un userv_addr;
      int userv_len,ucli_len;

      usfd = socket(AF_UNIX, SOCK_DGRAM, 0);

      if(usfd==-1)
      perror("\nsocket  ");

      bzero(&userv_addr,sizeof(userv_addr));
      userv_addr.sun_family = AF_UNIX;
      strcpy(userv_addr.sun_path, ADDRESS);

      userv_len = sizeof(userv_addr);

      fgets( buffer , 256 , stdin );
      sendto(sfd , buffer , 256 , 0 , ( struct sockaddr * ) &userv_addr ,
userv_len);
```

```
      recvfrom(sfd , buffer , 256 , 0 , ( struct sockaddr * ) &userv_addr
, & userv_len );
```

SOCKET PAIR   (usage -: "./a.out")

```
      int usfd[2];
      if(socketpair(AF_UNIX,SOCK_STREAM,0,usfd)==-1)
      perror("socketpair ");

      int c=fork();

      if(c==-1)
      perror("\nfork ");

      else if(c>0)
      {
            close(usfd[1]);
      }

      else if(c==0)
      {
            close(usfd[0]);
            dup2(usfd[1],0);
            execvp(file_name,args);
      }
```