Socket Programming

Giulio Grassi
giulio.grassi@lip6.fr

Exercise

- Not mandatory. **Highly suggested**, especially for those who never used socket

Exercise 1

- TCP server wait for messages from a client
- TCP Client receives input from user (a string) and sends the message to the server
- The server prints the message (remember: you might need multiple read to receive the entire message), reverts the message (optional) and sends back the message to the client
- The client prints the message

Exercise 2

- Try this out with UDP communication too
- Try Exercise 1 and 2 with and without select

Solution (C) ex1 with select - Client

```
int main(int argc, char *argv[])
{
   int sockfd, portno, n;
   struct sockaddr_in serv_addr;
   struct hostent *server;

   char buffer[256];
   if (argc < 3) {
      fprintf(stderr,"usage %s hostname port\n", argv[0]);
      exit(0);
   }
   portno = atoi(argv[2]);

   //creating the socket
   sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
   if (sockfd < 0)
      perror("ERROR opening socket");
   server = gethostbyname(argv[1]);
   if (server == NULL) {
      fprintf(stderr,"ERROR, no such host\n");
      exit(0);
   }
   bzero((char *) &serv_addr, sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   bcopy((char *)server->h_addr,
        (char *)&serv_addr.sin_addr.s_addr,
        server->h_length);
   serv_addr.sin_port = htons(portno);
```

Solution (C) ex1 with select - Client

```
//connecting to the server
  if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
     error("ERROR connecting");
  printf("Please enter the message: ");
  bzero(buffer,256);
  fgets(buffer,255,stdin);

  //sending a message to the server
  n = write(sockfd,buffer,strlen(buffer));
  printf ("msg : %s %d", buffer, n);
  fflush(stdout);

  if (n < 0)
      error("ERROR writing to socket");
  printf("Msg sent to the server %d", n);
  int dataSent = n; //later on, we will have to receive dataSent bytes from the server

  //reading the server's reply (using select)

  fd_set read_fd;
  int maxfd = sockfd;

  bzero(buffer,256);
  int receivedData = 0;
```

Solution (C) ex1 with select - Client

```
while (receivedData < dataSent) {
    FD_ZERO(&read_fd);
    FD_SET(sockfd, &read_fd);
    int result = select(maxfd + 1, &read_fd, NULL, NULL, NULL);
    if(result==-1){
      printf("ERROR select failed %d %s", errno, strerror(errno));
      close(sockfd);
      return -1;
    }
    if (result==0){
      //that's impossible because we didn't set any timer
    } else {
      if(FD_ISSET(sockfd, &read_fd)){
        //socket ready to be read
        n = read(sockfd,&buffer[receivedData],255-receivedData);
        if (n < 0){
          error("ERROR reading from socket");
        }else if (n==0){
          printf("The other endpoint has closed the communication\n");
          close(sockfd);
          return 0;
        }
        receivedData += n;
      }
      // if more file descriptors are available, check them here
    }
  }
  printf("%s\n",buffer);
  close(sockfd);
  return 0;
```

Solution (C) ex1 with select - Server

```
int main(int argc, char *argv[])
{
   int sockfd, newsockfd, portno;
   socklen_t clilen;
   char buffer[256];
   struct sockaddr_in serv_addr, cli_addr;
   int n;
   if (argc < 2) {
      fprintf(stderr,"ERROR, no port provided\n");
      exit(1);
   }

   //opening socket
   sockfd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
   if (sockfd < 0)
      perror("ERROR opening socket");

   bzero((char *) &serv_addr, sizeof(serv_addr));
   portno = atoi(argv[1]);
   serv_addr.sin_family = AF_INET;
   serv_addr.sin_addr.s_addr = INADDR_ANY;
   serv_addr.sin_port = htons(portno);
```

Solution (C) ex1 with select - Server

```
/* avoid EADDRINUSE error on bind() */
  int OptVal = 1;
  int res = setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, (char *)&OptVal, sizeof(OptVal));
  if (res == SOCKET_ERROR)  {
     error("setsockopt() SO_REUSEADDR failed, Err: %d %s \n", errno,strerror(errno));
  }

  if (bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0)
     error("ERROR on binding");

  listen(sockfd,2);
  clilen = sizeof(cli_addr);

  fd_set read_fd;
  int client[] = {-1,-1,-1,-1};
  int maxNumClient = 4;

  int numberOfClientServed = 0;
```

Solution (C) ex1 with select - Server

```
while(numberOfClientServed<5){
    FD_ZERO(&read_fd);
    FD_SET(sockfd, &read_fd);

    int maxfd = sockfd;
    int i;
    for (i=0; i< maxNumClient; i++){
      if (client[i]!=-1){
        FD_SET(client[i], &read_fd);
        if (maxfd<client[i]){
          maxfd = client[i];
        }
      }
    }
    int result = select(maxfd + 1, &read_fd, NULL, NULL, NULL);
    if(result==-1){
      printf("ERROR select failed %d %s", errno, strerror(errno));
      close(sockfd);
      return -1;
    }
    if (result==0){
      //that's impossible because we didn't set any timer
    } else {
      .....
```

Solution (C) ex1 with select - Server

```
….. (   else {   )
        if(FD_ISSET(sockfd, &read_fd)){
           printf("New connection request\n");
           newsockfd = accept(sockfd,  (struct sockaddr *) &cli_addr,  &clilen);
           if (newsockfd < 0)
              error("ERROR on accept");

           int newClientIndex = -1;
           for (i=0; i< maxNumClient; i++){
              if (client[i]==-1){
                 newClientIndex = i;
                 break;
              }
           }
           if(newClientIndex!=-1){
              client[newClientIndex] = newsockfd;
           } else {
              printf("The server cannot accept more clients\n");
              close(newsockfd);
           }
        }
```

Solution (C) ex1 with select - Server

```
for (i=0; i< maxNumClient; i++){
        if (client[i]!=-1){
          if(FD_ISSET(client[i], &read_fd)){
            //received data from client
            bzero(buffer,256);
            n = read(client[i],buffer,255);
            if (n < 0){
              error("ERROR reading from socket");
              close(client[i]);
              client[i]=-1;
            } else if (n==0){
              printf("Client has closed the connection\n");
              close(client[i]);
              client[i]=-1;
              numberOfClientServed++;
            } else {
              printf("Here is the message: %s %d\n",buffer, n);

              // ???? sending data is this way might cause a problem. What's the problem?
              n = write(client[i],buffer,n);

              if (n<0){
                error("ERROR sending data to client");
              } else {
                printf("Msg sent to the client %d \n", n);
              }
            }  }  }  }  }  //… closing all the open sockets
```