ITT440 NETWORK PROGRAMMING

C
Socket Programming

Basic Knowledge

- C is a powerful general-purpose programming language.
- The BSD sockets API is written in the C programming language.
- Most other programming languages provide similar interfaces, typically written as a wrapper library based on the C API.

Basic Function Used

Establishing a client-server through C socket programming involves the following functions:

- socket()
- bind()
- listen()
- accept()
- connect()
- send() / sendto()
- recv() / recvfrom()
- read() & write()
- close()

server-tcp.c

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int main(void)
{
  struct sockaddr_in sa;
  int SocketFD = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (SocketFD == -1) {
    perror("cannot create socket");
    exit(EXIT_FAILURE);
  }

  memset(&sa, 0, sizeof sa);

  sa.sin_family = AF_INET;
  sa.sin_port = htons(1100);
  sa.sin_addr.s_addr = htonl(INADDR_ANY);
```

```c
  if (bind(SocketFD,(struct sockaddr *)&sa, sizeof sa) == -1) {
    perror("bind failed");
    close(SocketFD);
    exit(EXIT_FAILURE);
  }

  if (listen(SocketFD, 10) == -1) {
    perror("listen failed");
    close(SocketFD);
    exit(EXIT_FAILURE);
  }

  for (;;) {
    int ConnectFD = accept(SocketFD, NULL, NULL);

    if (0 > ConnectFD) {
      perror("accept failed");
      close(SocketFD);
      exit(EXIT_FAILURE);
    }

    /* perform read write operations ...
    read(ConnectFD, buff, size)
    */

    if (shutdown(ConnectFD, SHUT_RDWR) == -1) {
      perror("shutdown failed");
      close(ConnectFD);
      close(SocketFD);
      exit(EXIT_FAILURE);
    }
    close(ConnectFD);
  }

  close(SocketFD);
  return EXIT_SUCCESS;
}
```

client-tcp.c

```c
void my_write(t_s *s)
{
    const char *message = "Hello, please enter your name: ";
    s->writeValue = write(s->acceptValue, message, strlen(message));
    check_error(s->writeValue, -1);
}

void my_read(t_s *s)
{
    bzero(s->buf,256);
    s->readValue = read(s->acceptValue, s->buf, BUF_SIZE);
    check_error(s->readValue, -1);
}
```

```c
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int main(void)
{
  struct sockaddr_in sa;
  int res;
  int SocketFD;

  SocketFD = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
  if (SocketFD == -1) {
    perror("cannot create socket");
    exit(EXIT_FAILURE);
  }

  memset(&sa, 0, sizeof sa);

  sa.sin_family = AF_INET;
  sa.sin_port = htons(1100);
  res = inet_pton(AF_INET, "192.168.1.3", &sa.sin_addr);

  if (connect(SocketFD, (struct sockaddr *)&sa, sizeof sa) == -1) {
    perror("connect failed");
    close(SocketFD);
    exit(EXIT_FAILURE);
  }

  /* perform read write operations ... */

  close(SocketFD);
  return EXIT_SUCCESS;
}
```

Exercises

By using C programming please create:

1. UDP time server and client that connect through port 22000.
2. A program using port 27679 TCP that passing multiple random numbers.
3. A UDP client-server port 11235 that passing Fibonacci series based on input from client where follow the following criteria:
   - x_n is term number "n"
   - The client must provide n.
   - The server must send term and onwards of five sequence.
   - The client can choose either want another five sequence or quit the program.
   - The server must continue send another five sequence onward of last term provided if ask to do so by the client.

References

- https://www.mathsisfun.com/numbers/fibonacci-sequence.htm
- https://www.badprog.com/c-tcp-ip-writing-and-reading-on-a-socket
- https://www.programiz.com/c-programming
- www.tutorialpoints.com
- Wikipedia
- Youtube