CENG393 Computer Networks
Labwork 5

1 Socket Programming: Handling Multiple Sockets

In this labwork, you will execute a server program which allows connections from multiple clients and forwards messages sent from these clients to all the other clients as well. Study the given codes both, then execute them as 1 server and multiple clients. Observe how the programs work.

1.1 Exercise

After executing the given programs, modify them to meet the following requirements:

- Whenever a client connects and disconnects, server must display a message on screen.
- When a client connects to a server, it must ask for a nickname from the user. Nickname must be appended to the beginning of every line read from the keyboard and this new string must be sent to the server.

1.1.1 Sample Run - Server

```
./server
127.0.0.1 connected.
192.168.1.16 connected.
127.0.0.1 disconnected.
192.168.1.21 connected.
. . .
```

1.1.2 Sample Run - Client #1

```
./client
Enter IP address of the Server
127.0.0.1
Enter your name: Kirk
Picard: Hello
Hi, whats up?
. . .
```

1.1.3 Sample Run - Client #2

```
./client
Enter IP address of the Server
192.168.1.1
Enter your name: Picard
Hello
Kirk: Hi, whats up?
. . .
```

1.2 select_server.c

```
#include <ctype.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <signal.h>
#include <stdio.h>

#define SIZE sizeof(struct sockaddr_in)
#define MAX 5

int client[MAX];
int ActiveClients = 0;

void findMax(int *maxfd) {
  int i;
  *maxfd = client[0];
  for (i = 1; i < MAX; i++)
    if (client[i] > *maxfd)
      *maxfd = client[i];
}

int main() {
  int sockfd, maxfd, nread, found, i, j;
  char buf[128];
  fd_set fds;
  struct sockaddr_in server = { AF_INET, 2000, INADDR_ANY };

  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    printf("Error creating SOCKET\n");
    return (0);
  }
  if (bind(sockfd, (struct sockaddr *) &server, SIZE) == -1) {
    printf("bind failed\n");
    return (0);
  }
  if (listen(sockfd, 5) == -1) {
    printf("listen failed\n");
    return (0);
  }

  findMax(&maxfd);

  for (;;) {
    findMax(&maxfd);
    maxfd = (maxfd > sockfd ? maxfd : sockfd) + 1;
    FD_ZERO(&fds);
    FD_SET(sockfd, &fds);
    for (i = 0; i < MAX; i++)
      if (client[i] != 0)
        FD_SET(client[i], &fds);

    /* Wait for some input or connection request. */
    select(maxfd, &fds, (fd_set *) 0, (fd_set *) 0, (struct timeval *) 0);

    /*If one of the clients has some input, read and send it to all others.*/
    for (i = 0; i < MAX; i++)
      if (FD_ISSET(client[i], &fds)) {
        nread = recv(client[i], buf, sizeof(buf), 0);
        /* If error or eof, terminate the connection */
        if (nread < 1) {
          close(client[i]);
          client[i] = 0;
          ActiveClients--;
        } else
          /* broadcast the message */
          for (j = 0; j < MAX; j++)
            if (client[j] != 0 && i != j)
```

```
              send(client[j], buf, nread, 0);
      }

    /* if there is a request for a new connection */
    if (FD_ISSET(sockfd, &fds)) {
      /* If no of active clients is less than MAX accept the request */
      if (ActiveClients < MAX) {
        found = 0;
        for (i = 0; i < MAX && !found; i++)
          if (client[i] == 0) {
            client[i] = accept(sockfd, NULL, NULL);
            found = 1;
            ActiveClients++;
          }
      }
    }
  }
}
```

1.3 select_client.c

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

#define SIZE sizeof(struct sockaddr_in)

int main() {
  int sockfd, nread;
  char buf[128], enter, resp;
  fd_set fds;
  char IP[20];
  struct sockaddr_in server = { AF_INET, 2000 };

  printf("\n\n\n\n\nEnter IP address of the Server\n");
  scanf("%s%c", IP, &enter);
  server.sin_addr.s_addr = inet_addr(IP);

  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    printf("Error creating SOCKET\n");
    return (0);
  }

  if (connect(sockfd, (struct sockaddr *) &server, SIZE) == -1) {
    printf("Connect failed\n");
    return (0);
  }

  printf("Enter a message (E to exit)\n");
  do {
    FD_ZERO(&fds);
    FD_SET(sockfd, &fds);
    FD_SET(0, &fds);
    /* Wait for some input. */
    select(sockfd + 1, &fds, (fd_set *) 0, (fd_set *) 0, (struct timeval *) 0);

    /* If either device has some input,read it and copy it to the other. */
    if (FD_ISSET(sockfd, &fds)) {
      nread = recv(sockfd, buf, sizeof(buf), 0);
      /* If error or eof, terminate. */
      if (nread < 1) {
```

```
        close(sockfd);
        exit(0);
      }
      buf[nread] = 0;
      printf("%s", buf);
    }

    if (FD_ISSET(0, &fds)) {
      nread = read(0, buf, sizeof(buf));
      /* If error or eof, terminate. */
      if (nread < 1) {
        close(sockfd);
        exit(0);
      } else if ((buf[0] == 'e' || buf[0] == 'E') && nread == 2) {
        close(sockfd);
        exit(0);
      } else
        send(sockfd, buf, nread, 0);
    }
  } while (1);
}
```