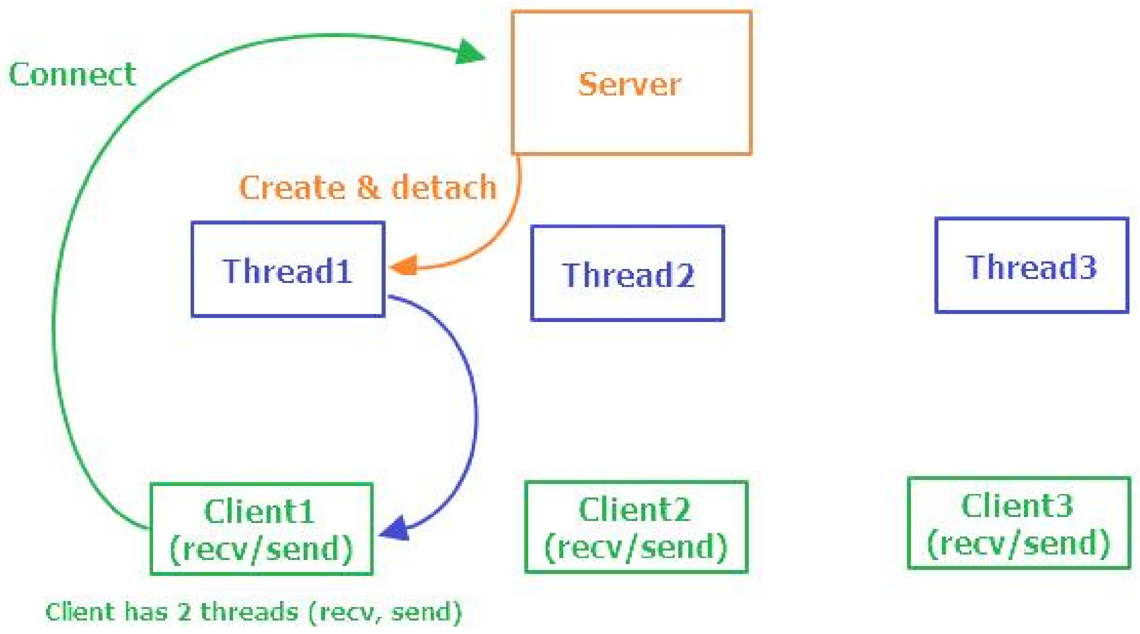
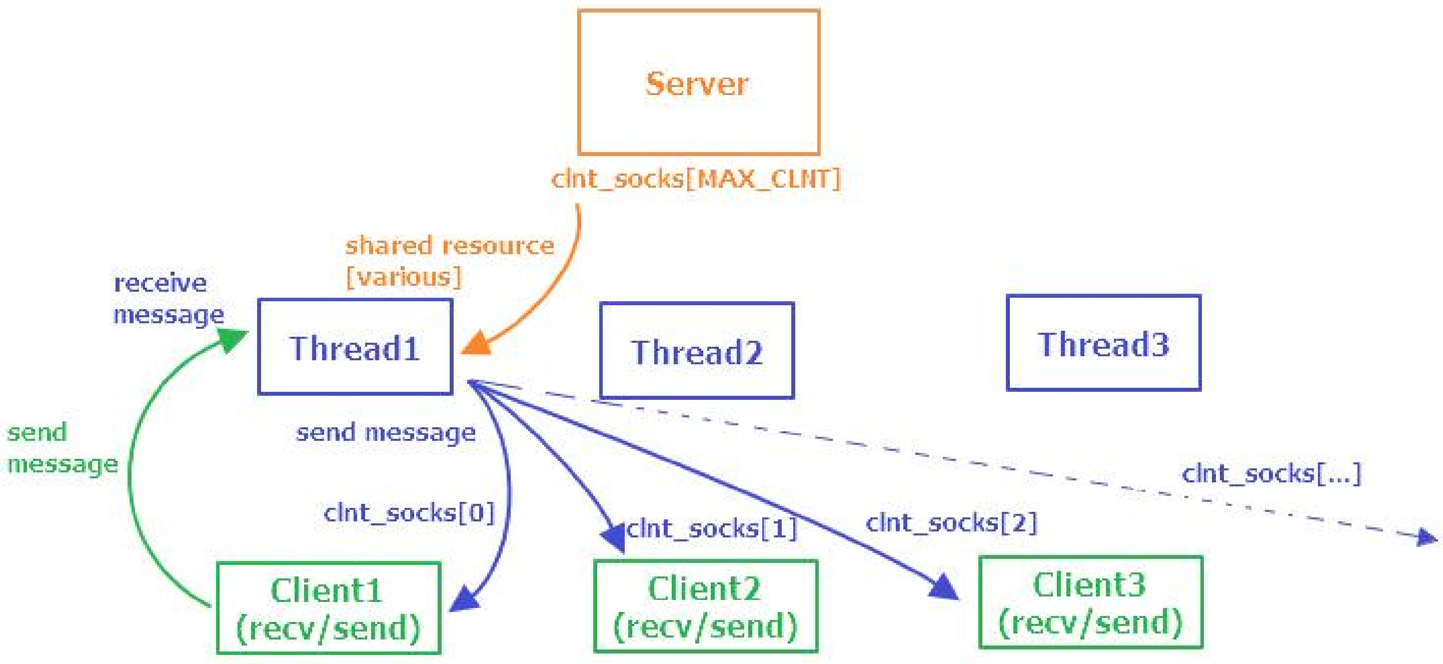
TI DSP, MCU, Xilinx Zynq FPGA Based Programming Expert Program

Instructor – Innova Lee (Sanghoon Lee)
gcccompil3r@gmail.com
Student – Howard Kim (Hyungju Kim)
mihaelkel@naver.com

Chatting Program





The server makes a thread whenever a client is connected. Each thread can send a message to all of the client connected with threads by referencing “clnt_socks[MAX_CLNT]”, which has a file descriptor of clients.

clients have 2 threads - fuctioning receive a message from their socket, and send a message to the socket. By separating the functions, can process “Non-blocking”.

chat_serv.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.h>
#include <time.h>
#define BUF_SIZE     128
#define MAX_CLNT     256
typedef struct sockaddr_in si;
typedef struct sockaddr *  sp;
int clnt_cnt = 0;
int clnt_socks[MAX_CLNT];
pthread_mutex_t mtx;
void err_handler(char* msg){
     fputs(msg, stderr);
     fputc('\n', stderr);
     exit(1);
}
void send_msg(char* msg,int len){
     int i;
     pthread_mutex_lock(&mtx);
     for(i=0;i<clnt_cnt;i++)
          write(clnt_socks[i], msg, len);
     pthread_mutex_unlock(&mtx);
}
void* clnt_handler(void* arg){
     int clnt_sock = *((int*)arg);
     int str_len = 0, i;
     char msg[BUF_SIZE];
     int cnt = 0;
     clock_t clk_start;
     double diff;
     char kick_msg[256]=“You’ve talked so much\n";
     clk_start = clock();
     while((str_len = read(clnt_sock, msg, sizeof(msg))) != 0){
          redo:
          diff = (double)(clock() - clk_start) / CLOCKS_PER_SEC;
          diff *= 20000;
          if((cnt > diff)&&(cnt > 4)){
               send_msg(kick_msg, strlen(kick_msg));
               goto tmp;
          }
          //initialize kick counter
          if(diff > 5){
               clk_start = clock();
               cnt = 0;
          }

          send_msg(msg, str_len);
          cnt++;
     }
tmp:
     sleep(3);
     goto redo;
     pthread_mutex_lock(&mtx);
```

```c
    for(i=0;i<clnt_cnt;i++){
        if(clnt_sock == clnt_socks[i]){
            while(i++ < clnt_cnt - 1)
                clnt_socks[i] = clnt_socks[i+1];
            break;
        }
    }
    clnt_cnt--;
    pthread_mutex_unlock(&mtx);
    close(clnt_sock);
    return NULL;
}
int main(int argc, char** argv){
    int serv_sock, clnt_sock;
    si serv_addr, clnt_addr;
    socklen_t addr_size;
    pthread_t t_id;
    if(argc != 2){
        printf("Usage : %s <port>\n",argv[0]);
        exit(1);
    }
    pthread_mutex_init(&mtx, NULL);
    serv_sock = socket(PF_INET, SOCK_STREAM, 0);
    if(serv_sock == -1)
        err_handler("socket() error");
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(atoi(argv[1]));
    if(bind(serv_sock, (sp)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("bind() error");
    if(listen(serv_sock, 30) == -1)
        err_handler("listen() error");
    for(;;){
        addr_size = sizeof(clnt_addr);
        clnt_sock = accept(serv_sock, (sp)&clnt_addr, &addr_size);
        pthread_mutex_lock(&mtx);
        clnt_socks[clnt_cnt++] = clnt_sock;
        pthread_mutex_unlock(&mtx);
        pthread_create(&t_id, NULL, clnt_handler, (void*)&clnt_sock);
        pthread_detach(t_id);
        printf("Connected Client IP : %s\n",inet_ntoa(clnt_addr.sin_addr));
    }
    close(serv_sock);
    return 0;
}
```

chat_clnt.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/epoll.h>

#define BUF_SIZE     128
#define NAME_SIZE     32

typedef struct sockaddr_in si;
typedef struct sockaddr * sp;

char name[NAME_SIZE] = "[DEFAULT]";
char msg[BUF_SIZE];

void err_handler(char* msg){
    fputs(msg, stderr);
    fputc('\n',stderr);
    exit(1);
}

void* send_msg(void* arg){
    int sock = *((int*)arg);
    char name_msg[NAME_SIZE + BUF_SIZE];

    for(;;){
        fgets(msg, BUF_SIZE, stdin);
        if(!strcmp(msg, "q\n") || !strcmp(msg, "Q\n")){
            close(sock);
            exit(0);
        }
        sprintf(name_msg, "%s %s", name, msg);
        write(sock, name_msg, strlen(name_msg));
    }
    return NULL;
}

void* recv_msg(void* arg){
    int sock = *((int*)arg);
    char name_msg[NAME_SIZE + BUF_SIZE];
    int str_len;

    for(;;){
        str_len = read(sock, name_msg, NAME_SIZE + BUF_SIZE - 1);
        if(str_len == -1)
            return (void*)-1;

        name_msg[str_len] = 0;
        fputs(name_msg, stdout);
    }
    return NULL;
}

int main(int argc,char** argv){
    int sock;
    si serv_addr;
    pthread_t snd_thread, rcv_thread;
    void* thread_ret;
    if(argc != 4){
        printf("Usage: %s <IP> <port><name>\n",argv[0]);
        exit(1);
    }
    sprintf(name, "[%s]", argv[3]);
    sock = socket(PF_INET, SOCK_STREAM, 0);

    if(sock == -1)
        err_handler("socket() error");

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));

    if(connect(sock, (sp)&serv_addr, sizeof(serv_addr)) == -1)
        err_handler("connect() error");

    pthread_create(&snd_thread, NULL, send_msg, (void*)&sock);
    pthread_create(&rcv_thread, NULL, recv_msg, (void*)&sock);

    pthread_join(snd_thread, &thread_ret);
    pthread_join(rcv_thread, &thread_ret);

    close(sock);
    return 0;
}

```