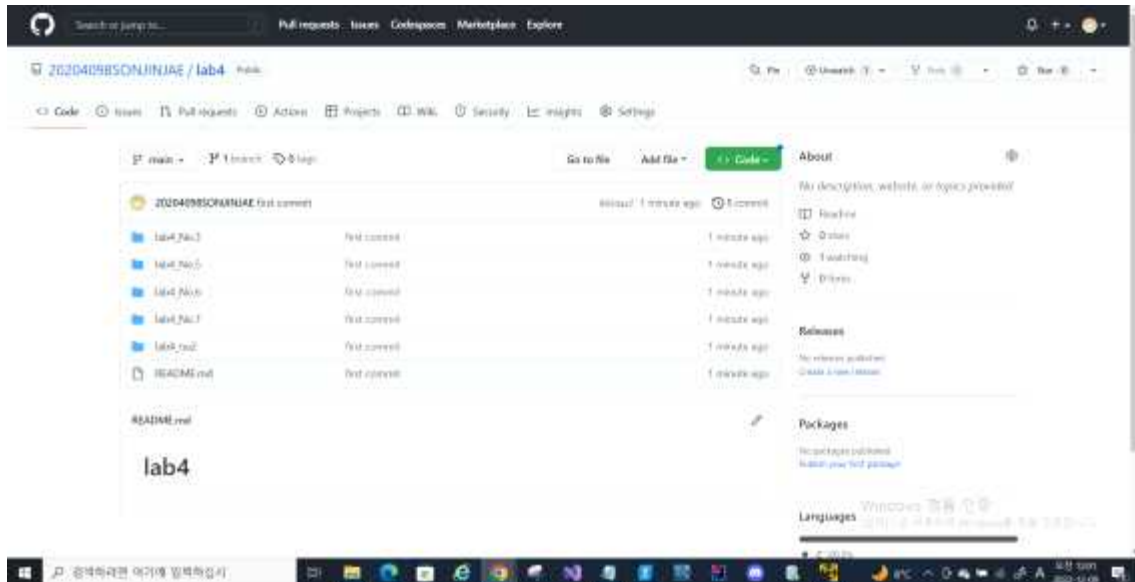


# 시스템 프로그래밍 lab 4 project

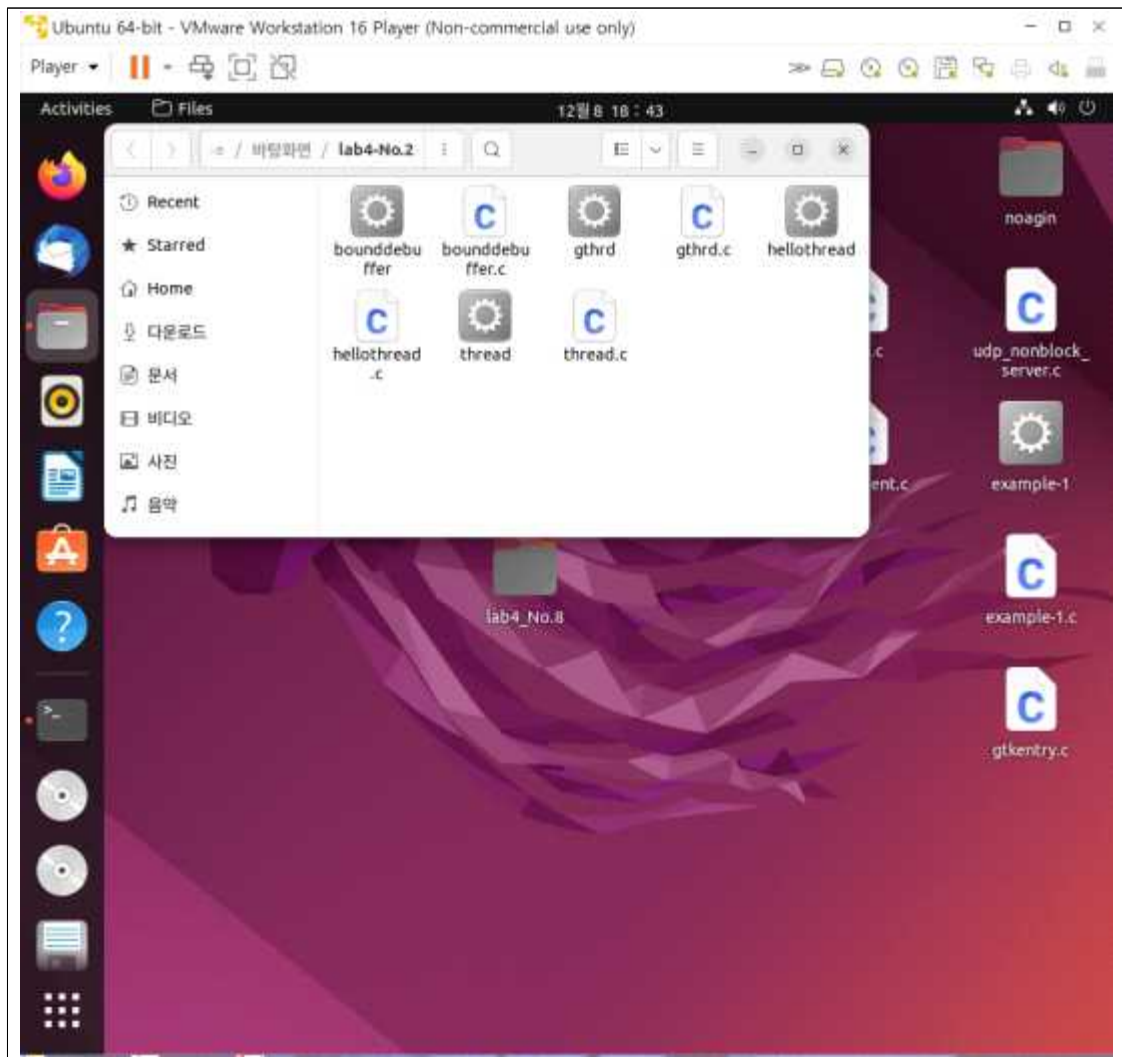


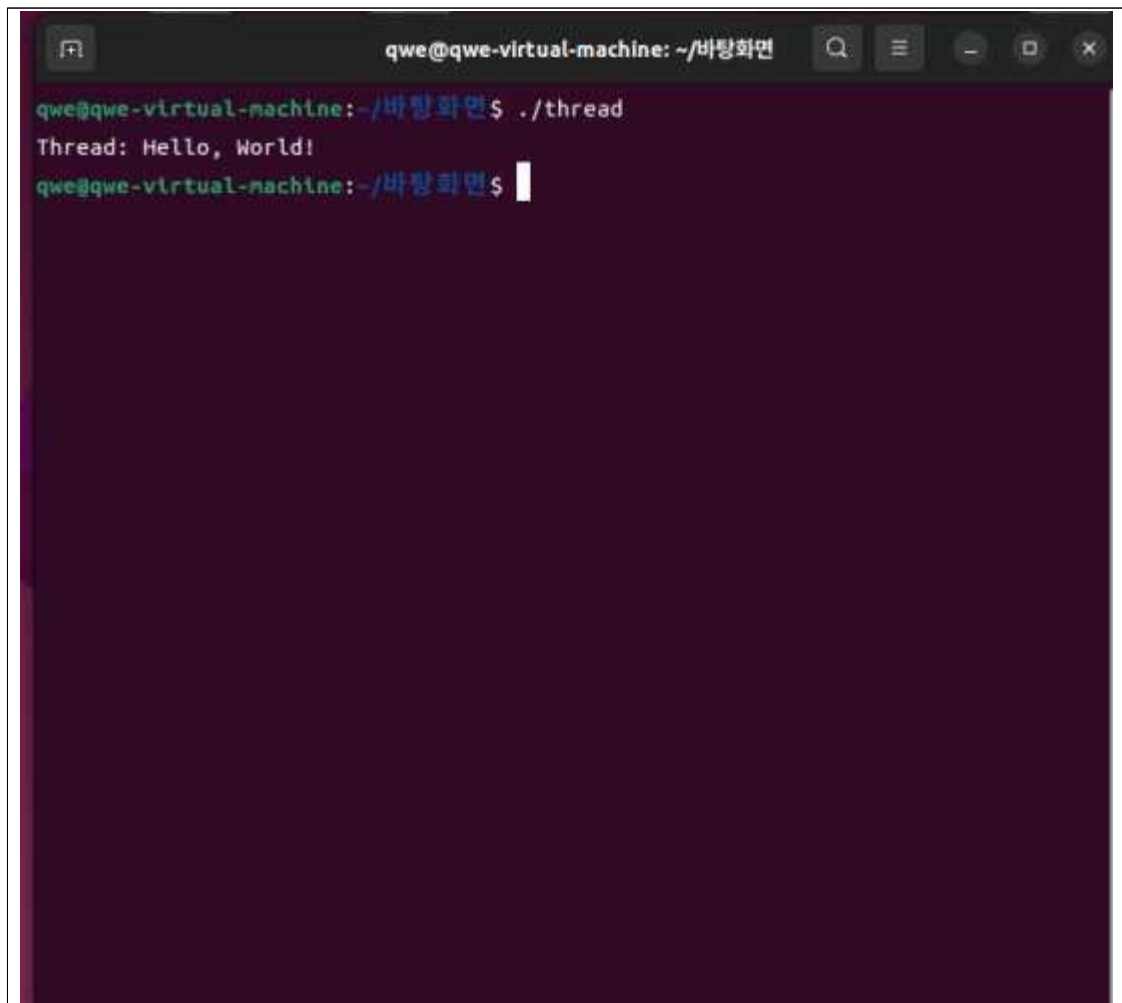
과목명	시스템 프로그래밍 (리눅스)
담당 교수명	김성우
학번	20204098
이름	손진제

1. 자신의 github 저장소에 lab4 프로젝트를 생성하고 아래의 모든 과제 프로그램을 업로드한다.



2. 쓰레드 관련 함수들을 사용하여 프로그램을 작성하고 실행하여 보고, 익숙해지도록 사용해 본다.

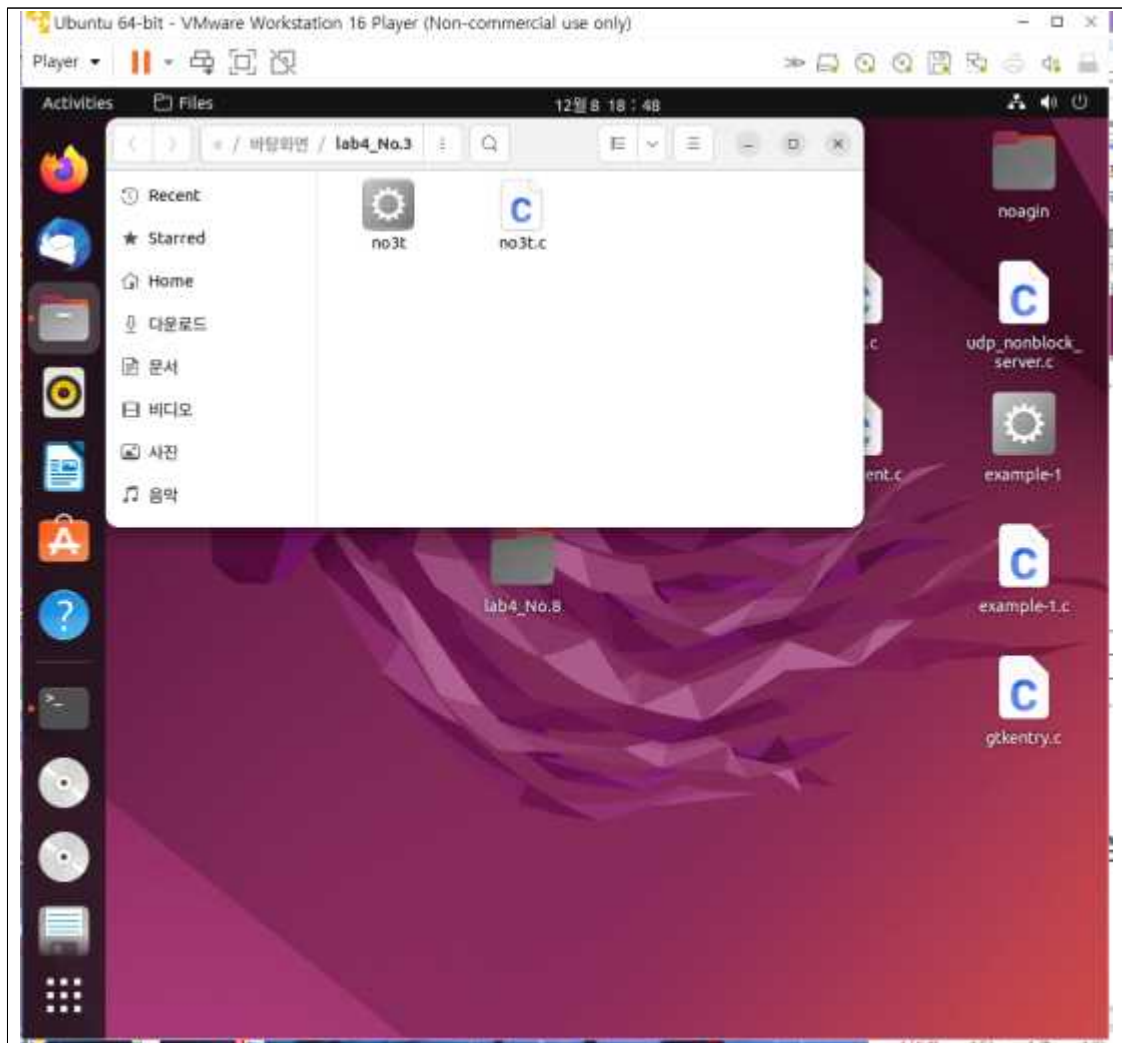


A terminal window titled 'qwe@qwe-virtual-machine: ~/바탕화면'. The prompt is 'qwe@qwe-virtual-machine:~/바탕화면\$'. The user has entered './thread' and the output is 'Thread: Hello, World!'. The prompt is now 'qwe@qwe-virtual-machine:~/바탕화면\$' with a cursor.

```
qwe@qwe-virtual-machine: ~/바탕화면
qwe@qwe-virtual-machine:~/바탕화면$ ./thread
Thread: Hello, World!
qwe@qwe-virtual-machine:~/바탕화면$
```

3. 스레드를 사용하여 생산자 소비자 문제를 해결하는 제한 버퍼 (Bounded Buffer)를 생성하고 활용하는 프로그램을 구현하십시오. 단, 생산자와 소비자 스레드는 각각 둘 이상 가능해야 한다.

```
qwe@qwe-virtual-machine: ~/바탕화면
Consumer 0 consumes: 90001
Consumer 0 consumes: 90002
Consumer 0 consumes: 90003
Consumer 0 consumes: 90004
Consumer 0 consumes: 90005
Consumer 0 consumes: 90006
Consumer 0 consumes: 90007
Consumer 0 consumes: 90008
Consumer 0 consumes: 90009
Consumer 0 consumes: 90010
Consumer 0 consumes: 90011
Consumer 0 consumes: 90012
Consumer 0 consumes: 90013
Consumer 0 consumes: 90014
Consumer 0 consumes: 90015
Consumer 0 consumes: 90016
Consumer 0 consumes: 90017
Consumer 0 consumes: 90018
Consumer 0 consumes: 90019
Consumer 0 consumes: 90020
Consumer 0 consumes: 90021
Consumer 0 consumes: 90022
Consumer 0 consumes: 90023
Consumer 0 consumes: 90024
```



4. 클라이언트(자식) 쓰레드들로부터 메시지 전송 요청을 받으면 서버(부모) 쓰레드는 모든 클라이언트 쓰레드에게 메시지를 발송하는 프로그램을 구현하시오.

(힌트: 소켓은 사용하지 말고 데이터 전송을 위한 동기화를 위해 뮤텍스와 조건변수를 사용한다.)

5. 소켓을 이용하여 프로그램을 작성하고 실행하여 보고, 익숙해지

도록 사용해 본다.

6. 멀티프로세스/쓰레드, select 또는 epoll 을 사용하여 다중 클라이언트를 처리하는 채팅 프로그램을 구현하시오.

no6.c 소스(1)

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<unistd.h>
4 #include<string.h>
5 #include<arpa/inet.h>
6 #include<sys/socket.h>
7 #include<netinet/in.h>
8 #include<pthread.h>
9 #include<time.h>
10
11 #define BUF_SIZE 100
12 #define MYPOR 3333
13 #define MAX_CLNT 100
14 #define MAX_IP 30
15
16 void *handle_clnt(void *arg);
17 void send_msg(char *msg, int len);
18 void error_handling(char *msg);
19 char *serverState(int count);
20
21 int clnt_cnt=0;
22 int clnt_socks[MAX_CLNT];
23 pthread_mutex_t mutx;
24
25 int main(int argc, char *argv[]){
26     int serv_sock, clnt_sock;
27     struct sockaddr_in serv_adr, clnt_adr;
28     int clnt_adr_sz;
29     pthread_t t_id;
30
31     /** time log **/
32     struct tm *t;
33     time_t timer = time(NULL);
34     t=localtime(&timer);
35
36     pthread_mutex_init(&mutx, NULL);
37     serv_sock=socket(PF_INET, SOCK_STREAM, 0);
38
39     memset(&serv_adr, 0, sizeof(serv_adr));
40     serv_adr.sin_family=AF_INET;
41     serv_adr.sin_addr.s_addr=htonl(INADDR_ANY);
42     serv_adr.sin_port=htons(MYPOR);
43
44     if (bind(serv_sock, (struct sockaddr*)&serv_adr, sizeof(serv_adr))==-1)
45         error_handling("bind() error");
46     if (listen(serv_sock, 5)==-1)
47         error_handling("listen() error");
```

no6.c 소스(2)

```

49     printf("Server start!!\n");
50
51     while(1){
52         t=localtime(&ttime);
53         clnt_addr_sz=sizeof(clnt_addr);
54         clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_sz);
55
56         pthread_mutex_lock(&mutex);
57         clnt_socks[clnt_cnt++]=clnt_sock;
58         pthread_mutex_unlock(&mutex);
59
60         pthread_create(&t_id, NULL, handle_clnt, (void*)&clnt_sock);
61         pthread_detach(t_id);
62         printf("Connected client IP : %s ", inet_ntoa(clnt_addr.sin_addr));
63         printf("(%d.%d.%d.%d:%d)\n", t->tm_year+1900, t->tm_mon+1, t->tm_mday,
64             t->tm_hour, t->tm_min);
65         printf("chatter (%d/100)\n", clnt_cnt);
66     }
67     close(serv_sock);
68     return 0;
69 }
70
71 void *handle_clnt(void *arg){
72     int clnt_sock=((int*)arg);
73     int str_len=0, i;
74     char msg[BUF_SIZE];
75
76     while((str_len=read(clnt_sock, msg, sizeof(msg)))!=0)
77         send_msg(msg, str_len);
78
79     // remove disconnected client
80     pthread_mutex_lock(&mutex);
81     for (i=0; i<clnt_cnt; i++)
82     {
83         if (clnt_sock==clnt_socks[i])
84         {
85             while(i++<clnt_cnt-1)
86                 clnt_socks[i]=clnt_socks[i+1];
87             break;
88         }
89     }
90     clnt_cnt--;
91     pthread_mutex_unlock(&mutex);
92     close(clnt_sock);

```

no6.c 소스(3)



```
no6.c
~/no6_client.c

15
16 while((str_len=read(clnt_sock, msg, sizeof(msg)))!=0)
17     send_msg(msg, str_len);
18
19 // remove disconnected client
20 pthread_mutex_lock(&mutex);
21 for (i=0; i<clnt_cnt; i++)
22 {
23     if (clnt_sock==clnt_socks[i])
24     {
25         while(i++<clnt_cnt-1)
26             clnt_socks[i]=clnt_socks[i+1];
27         break;
28     }
29 }
30 clnt_cnt--;
31 pthread_mutex_unlock(&mutex);
32 close(clnt_sock);
33 return NULL;
34 }
35
36 void send_msg(char* msg, int len){
37     int i;
38     pthread_mutex_lock(&mutex);
39     for (i=0; i<clnt_cnt; i++)
40         write(clnt_socks[i], msg, len);
41     pthread_mutex_unlock(&mutex);
42 }
43
44 void error_handling(char *msg){
45     fputs(msg, stderr);
46     fputc('\n', stderr);
47     exit(1);
48 }
49
50 char* serverState(int count){
51     char* stateMsg = malloc(sizeof(char) * 20);
52     strcpy(stateMsg, "None");
53
54     if (count < 5)
55         strcpy(stateMsg, "Good");
56     else
57         strcpy(stateMsg, "Bad");
58
59     return stateMsg;
60 }
61
```

no6\_client.c 소스 (1)

```
no6.c
~/no6_client.c

1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<unistd.h>
4 #include<string.h>
5 #include<arpa/inet.h>
6 #include<sys/socket.h>
7 #include<pthread.h>
8 #include<time.h>
9
10 #define BUF_SIZE 100
11 #define NORMAL_SIZE 20
12 #define MYPORT 3333
13
14 void* send_msg(void* arg);
15 void* rcv_msg(void* arg);
16 void error_handling(char* msg);
17
18 char name[NORMAL_SIZE] = {0}; // name
19 char msg_form[NORMAL_SIZE]; // msg form
20 char serv_time[NORMAL_SIZE]; // server time
21 char msg[BUF_SIZE]; // msg
22 char serv_port[NORMAL_SIZE]; // server port number
23 char cint_ip[NORMAL_SIZE]; // client ip address
24
25
26 int main(int argc, char *argv[])
27 {
28     int sock;
29     struct sockaddr_in serv_addr;
30     pthread_t snd_thread, rcv_thread;
31     void* thread_return;
32
33     if (argc != 3)
34     {
35         printf("usage : %s <ip> <name>\n", argv[0]);
36         exit(1);
37     }
38
39     /** local time */
40     struct tm *t;
41     time_t timer = time(NULL);
42     t = localtime(&timer);
43     sprintf(serv_time, "%d %d %d %d %d", t->tm_year+1900, t->tm_mon+1, t->tm_mday, t->tm_hour,
44             t->tm_min);
45
46     sprintf(name, "[%s/%s]", argv[2], argv[1]);
47     sprintf(cint_ip, "%s", argv[1]);
```

no6\_client.c 소스 (2)

```

48 sock=socket(PF_INET, SOCK_STREAM, 0);
49
50 memset(&serv_addr, 0, sizeof(serv_addr));
51 serv_addr.sin_family=AF_INET;
52 serv_addr.sin_addr.s_addr=inet_addr(argv[1]);
53 serv_addr.sin_port=htons(MYPORT);
54
55 if (connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))!=-1)
56     error_handling(" connect() error");
57
58 printf("chatting Start!!!\n");
59
60 pthread_create(&snd_thread, NULL, send_msg, (void*)&sock);
61 pthread_create(&rcv_thread, NULL, rcv_msg, (void*)&sock);
62 pthread_join(snd_thread, &thread_return);
63 pthread_join(rcv_thread, &thread_return);
64 close(sock);
65 return 0;
66 }
67
68 void* send_msg(void* arg)
69 {
70     int sock=((int*)arg);
71     char name_msg[NORMAL_SIZE+BUF_SIZE];
72     char myInfo[BUF_SIZE];
73     char* who = NULL;
74     char temp[BUF_SIZE];
75
76     /** send join message **/
77     sprintf(myInfo, "%s join. \n", name);
78     write(sock, myInfo, strlen(myInfo));
79
80     while(1)
81     {
82         fgets(msg, BUF_SIZE, stdin);
83
84         sprintf(name_msg, "%s %s", name, msg);
85         write(sock, name_msg, strlen(name_msg));
86     }
87     return NULL;
88 }
89
90 void* rcv_msg(void* arg)
91 {

```

no6\_client.c 소스(3)

```

92     int sock=((int*)arg);
93     char name_msg[NORMAL_SIZE+BUF_SIZE];
94     int str_len;
95
96     while(1)
97     {
98         str_len=read(sock, name_msg, NORMAL_SIZE+BUF_SIZE-1);
99         if (str_len==-1)
100             return (void*)-1;
101         name_msg[str_len]=0;
102         fputs(name_msg, stdout);
103     }
104     return NULL;
105 }
106
107 void error_handling(char* msg)
108 {
109     fputs(msg, stderr);
110     fputc('\n', stderr);
111     exit(1);
112 }

```

no6\_server.c 소스(1)

```
no6_kc
no6_server.c
Save
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<unistd.h>
4 #include<string.h>
5 #include<arpa/inet.h>
6 #include<sys/socket.h>
7 #include<netinet/in.h>
8 #include<pthread.h>
9 #include<time.h>
10
11 #define BUF_SIZE 100
12 #define MYPORT 3333
13 #define MAX_CLNT 100
14 #define MAX_IP 30
15
16 void *handle_clnt(void *arg);
17 void send_msg(char *msg, int len);
18 void error_handling(char *msg);
19 char *serverState(int count);
20
21 int clnt_cnt=0;
22 int clnt_socks[MAX_CLNT];
23 pthread_mutex_t mutex;
24
25 int main(int argc, char *argv[]){
26     int serv_sock, clnt_sock;
27     struct sockaddr_in serv_addr, clnt_addr;
28     int clnt_addr_sz;
29     pthread_t t_id;
30
31     /** time log **/
32     struct tm *t;
33     time_t timer = time(NULL);
34     t=localtime(&timer);
35
36
37     pthread_mutex_init(&mutex, NULL);
38     serv_sock=socket(PF_INET, SOCK_STREAM, 0);
39
40     memset(&serv_addr, 0, sizeof(serv_addr));
41     serv_addr.sin_family=AF_INET;
42     serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
43     serv_addr.sin_port=htons(MYPORT);
44
45     if (bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))<0)
```

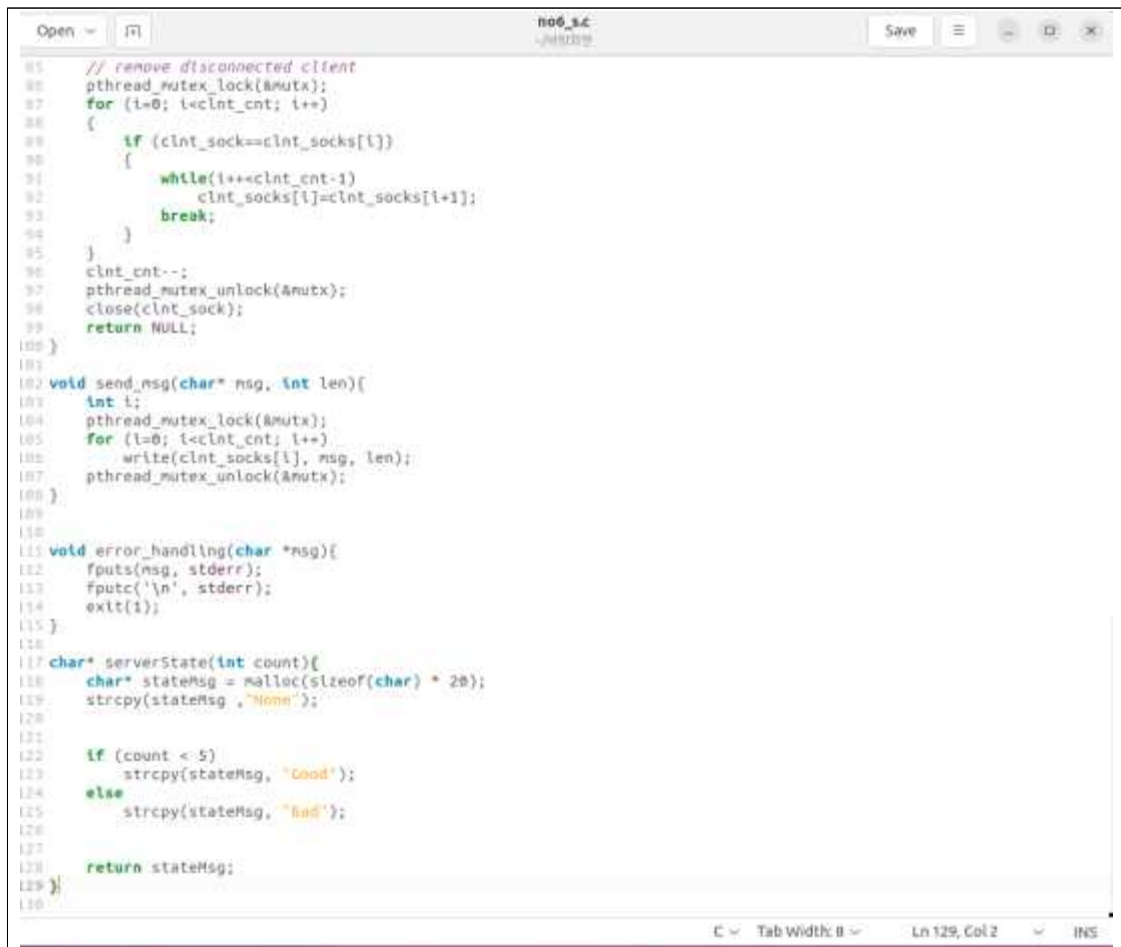
no6\_server소스(2)

```

43
44 if (bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))<0)
45     error_handling("bind() error");
46 if (listen(serv_sock, 5)<0)
47     error_handling("listen() error");
48
49 printf("Server start!\n");
50
51 while(1){
52     t=localtime(&timer);
53     clnt_addr_sz=sizeof(clnt_addr);
54     clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_addr_sz);
55
56     pthread_mutex_lock(&mutex);
57     clnt_socks[clnt_cnt++]=clnt_sock;
58     pthread_mutex_unlock(&mutex);
59
60     pthread_create(&t_id, NULL, handle_clnt, (void*)&clnt_sock);
61     pthread_detach(t_id);
62     printf("Connected client IP : %s\n", inet_ntoa(clnt_addr.sin_addr));
63     printf("%d %d %d %d\n", t->tm_year+1900, t->tm_mon+1, t->tm_mday,
64           t->tm_hour, t->tm_min);
65     t->tm_hour, t->tm_min);
66     printf("chatter (%d,%d)\n", clnt_cnt);
67 }
68 close(serv_sock);
69 return 0;
70 }
71
72 void *handle_clnt(void *arg){
73     int clnt_sock=((int*)arg);
74     int str_len=0, l;
75     char msg[BUF_SIZE];
76
77     while((str_len=read(clnt_sock, msg, sizeof(msg)))>0)
78         send_msg(msg, str_len);
79
80     // remove disconnected client
81     pthread_mutex_lock(&mutex);
82     for (l=0; l<clnt_cnt; l++)
83     {
84         if (clnt_sock==clnt_socks[l])
85             break;
86     }
87 }

```

no6\_server.c 소스(3)



```
105 // remove disconnected client
106 pthread_mutex_lock(&mutx);
107 for (i=0; i<clnt_cnt; i++)
108 {
109     if (clnt_sock==clnt_socks[i])
110     {
111         while(i++<clnt_cnt-1)
112             clnt_socks[i]=clnt_socks[i+1];
113         break;
114     }
115 }
116 clnt_cnt--;
117 pthread_mutex_unlock(&mutx);
118 close(clnt_sock);
119 return NULL;
120 }
121
122 void send_msg(char* msg, int len){
123     int i;
124     pthread_mutex_lock(&mutx);
125     for (i=0; i<clnt_cnt; i++)
126         write(clnt_socks[i], msg, len);
127     pthread_mutex_unlock(&mutx);
128 }
129
130 void error_handling(char *msg){
131     fputs(msg, stderr);
132     fputc('\n', stderr);
133     exit(1);
134 }
135
136 char* serverState(int count){
137     char* stateMsg = malloc(sizeof(char) * 20);
138     strcpy(stateMsg, "None");
139
140     if (count < 5)
141         strcpy(stateMsg, "Good");
142     else
143         strcpy(stateMsg, "Bad");
144
145     return stateMsg;
146 }
```

7. TCP 소켓을 이용하여 HTTP GET 및 POST 메소드 및 CGI 프로그램 실행을 구현하는 간단한 웹서버를 구현하시오.

(힌트: POST 메소드의 형식은 다음 링크를 참고하세요.

<https://developer.mozilla.org/ko/docs/Web/HTTP/Methods/POST>)

no7.c 소스 (1)

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <errno.h>
4 #include <string.h>
5 #include <fcntl.h>
6 #include <sys/socket.h>
7 #include <arpa/inet.h>
8 #define PORT 3490 /* default port */
9 #define QLEN 10 /* request queue size */
10 #define BUF_SIZE 1024
11
12 void *handle_clnt(int sockfd);
13 void send_err(int sockfd);
14 void send_msg(int sockfd);
15
16 int main(int argc, char *argv[])
17 {
18     int sockfd, new_fd; /* listen on sock_fd, new
19                          connection on new_fd */
20     struct sockaddr_in server_addr; /* structure to
21                                     hold server's address */
22
23     struct sockaddr_in client_addr; /* structure to hold
24                                     client's address */
25     int alen; /* length of address */
26     fd_set readfds, activefds; /* the set of read descriptors */
27     int i, maxfd = 0, numbytes;
28     char buf[100];
29     if ((sockfd = socket (AF_INET, SOCK_STREAM, 0)) < 0) {
30         perror("socket() failed");
31         exit(1);
32     }
33     memset(&server_addr, 0, sizeof(server_addr));
34     server_addr.sin_family = AF_INET;
35     server_addr.sin_port = htons(PORT);
36     server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
37     if (bind (sockfd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
38         perror("bind() failed");
39         exit(1);
40     }
41     /* Specify a size of request queue */
42     if (listen(sockfd, QLEN) < 0) {
43         fprintf(stderr, "listen failed\n");
44         exit(1);
45     }
46     alen = sizeof(client_addr);
47     /* Initialize the set of active sockets. */

```

no7.c 소스 (2)

```

48 FD_ZERO (&activefds);
49 FD_SET (sockfd, &activefds);
50 maxfd = sockfd;
51 /* Main server loop - accept and handle
52 requests */
53 fprintf( stderr, "Server up and running.\n");
54 while (1) {
55     printf("SERVER: Waiting for contact..., %d\n", maxfd);
56     /* Block until input arrives on one or more
57        active sockets. */
58     readfds = activefds;
59     if(select (maxfd + 1, &readfds, NULL, NULL, NULL) < 0)
60     {
61         perror ("select");
62         exit (EXIT_FAILURE);
63     }
64     /* Service all the sockets with input pending. */
65     for (i = 0; i <= maxfd; i++) {
66         if (FD_ISSET (i, &readfds)) {
67             if (i == sockfd) {
68                 if ( (new_fd=accept(sockfd, (struct sockaddr *)&client_addr, &alen)) < 0) {
69                     fprintf(stderr, "accept failed\n");
70                     exit (1);
71                 }
72                 FD_SET(new_fd, &activefds); //add the new socket desc to our active connections set
73                 if (new_fd > maxfd)
74                     maxfd = new_fd;
75             }
76             else {
77                 printf("handle client\n");
78                 handle_client(i);
79             }
80             close(i);
81             FD_CLR(i, &activefds);
82         }
83     }
84 }
85 close(sockfd);
86 }
87
88 void *handle_client(int client_sock)
89 {
90     int i;
91     int recv=0, str_len=0;
92     int readcnt=0; // read count

```

### no7.c 소스 (3)

```

93 char method[16];
94 if ((str_len=read(client_sock, &method, MAX_SIZE)) == -1) {
95     printf("read: error\n");
96     exit(1);
97 }
98 recv += str_len;
99 strcpy(method, strtok(msg, " "));
100 if (strcmp(method, "GET") != 0)
101     send_err(client_sock);
102 else
103     send_msg(client_sock); // 메시지 전송(이름)
104 return NULL;
105 }
106 void send_err(int client_sock) // send to all
107 {
108     char protocol[] = "HTTP/1.1 400-Bad Request\r\n";
109     char server[] = "Server:Netstack-Enterprise 0.0.1\r\n";
110     char contenttype[] = "Content-type:text/html\r\n";
111     char html[] = "<html><head><title>Bad Request</title></head><body><div>Bad Request</div></body></html>\r\n";
112     char end[] = "\r\n";
113     printf("send err\n");
114     write(client_sock, protocol, strlen(protocol));
115     write(client_sock, server, strlen(server));
116     write(client_sock, contenttype, strlen(contenttype));
117     write(client_sock, end, strlen(end));
118     write(client_sock, html, strlen(html));
119     fflush(fdopen(client_sock, "w"));
120 }
121 void send_msg(int client_sock) // send to all
122 {
123     char protocol[] = "HTTP/1.1 200-OK\r\n";
124     char server[] = "Server:Netstack-Enterprise 0.0.1\r\n";
125     char contentlength[] = "Content-length: 130\r\n";
126     char contenttype[] = "Content-type:text/html\r\n";
127     char html[] = "<html><head><title>Server</title></head><body><div>Hello and testing!</div></body></html>\r\n";
128     char end[] = "\r\n";
129     printf("send msg: len=130", strlen(html));
130     write(client_sock, protocol, strlen(protocol));
131     write(client_sock, server, strlen(server));
132     write(client_sock, contentlength, strlen(contentlength));
133     write(client_sock, contenttype, strlen(contenttype));
134     write(client_sock, end, strlen(end));
135     write(client_sock, html, strlen(html));
136     fflush(fdopen(client_sock, "w"));
137 }

```

### no7\_server.c 소스(1)



```

1: #include <stdio.h>
2: #include <stdlib.h>
3: #include <errno.h>
4: #include <string.h>
5: #include <sys/types.h>
6: #include <netinet/in.h>
7: #include <sys/socket.h>
8: #include <arpa/inet.h>
9: #include <sys/wait.h>
10:
11: #define NYPOR 3490
12: #define BACKLOG 10
13: #define MAXBUF 100
14:
15: main() {
16:     /* 클라이언트 소켓 문법 획득 */
17:     int ssock, csock;
18:     struct sockaddr_in serv_addr;
19:     struct sockaddr_in cInt_addr;
20:     char buf[MAXBUF];
21:     int sin_size;
22:
23:     if ((ssock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
24:         perror("socket");
25:         exit(1);
26:     }
27:
28:     memset(&serv_addr, 0, sizeof(serv_addr));
29:     serv_addr.sin_family = AF_INET;
30:     serv_addr.sin_port = htons(NYPOR);
31:     serv_addr.sin_addr.s_addr = INADDR_ANY;
32:     if (bind(ssock, (struct sockaddr *) &serv_addr, sizeof(struct sockaddr)) == -1) {
33:         perror("bind");
34:         exit(1);
35:     }
36:
37:     if (listen(ssock, BACKLOG) == -1) {
38:         perror("listen");
39:         exit(1);
40:     }
41:
42:     while (1) {
43:         sin_size = sizeof(struct sockaddr_in);
44:         if ((csock = accept(ssock, (struct sockaddr *) &cInt_addr, &sin_size)) == -1) {
45:             perror("accept");
46:             continue;
47:         }
48:         printf("server : got connection from %s\n", inet_ntoa(cInt_addr.sin_addr));

```

no7\_server.c (2) 소스

```

48:     memset(buf, 0, MAXBUF);
49:
50:     if (recv(csock, buf, MAXBUF, 0) == -1) {
51:         perror("recv");
52:         exit(0);
53:     }
54:
55:     printf("%s\n", buf);
56:
57:     if (send(csock, buf, strlen(buf), 0) == -1) {
58:         perror("send");
59:         close(csock);
60:         exit(0);
61:     }
62:     close(csock);
63: }
64: }

```

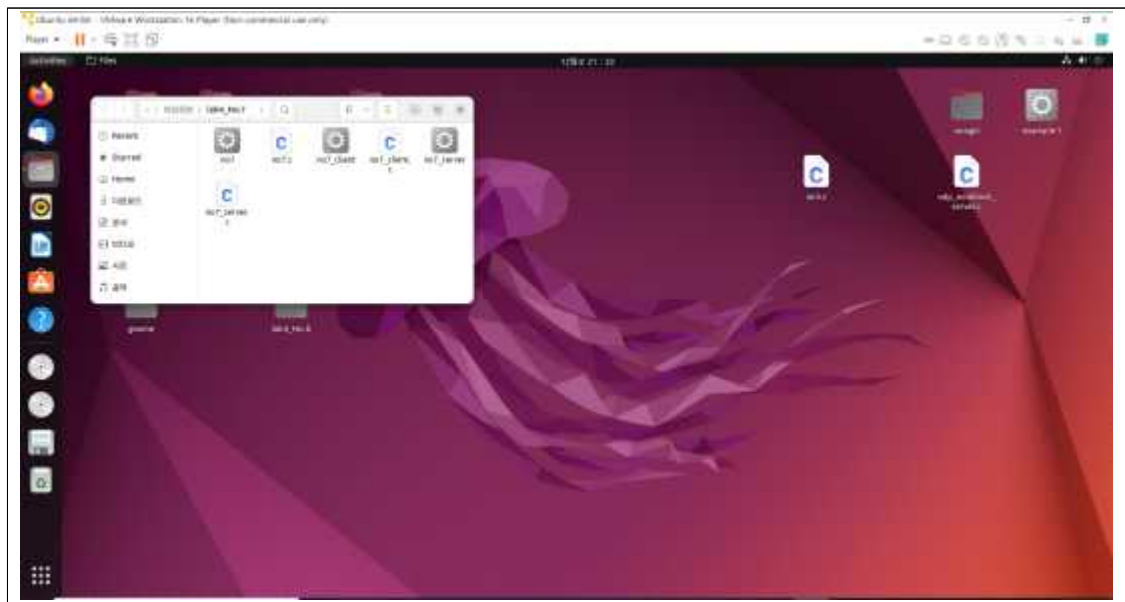
no7\_client.c (1) 소스

```

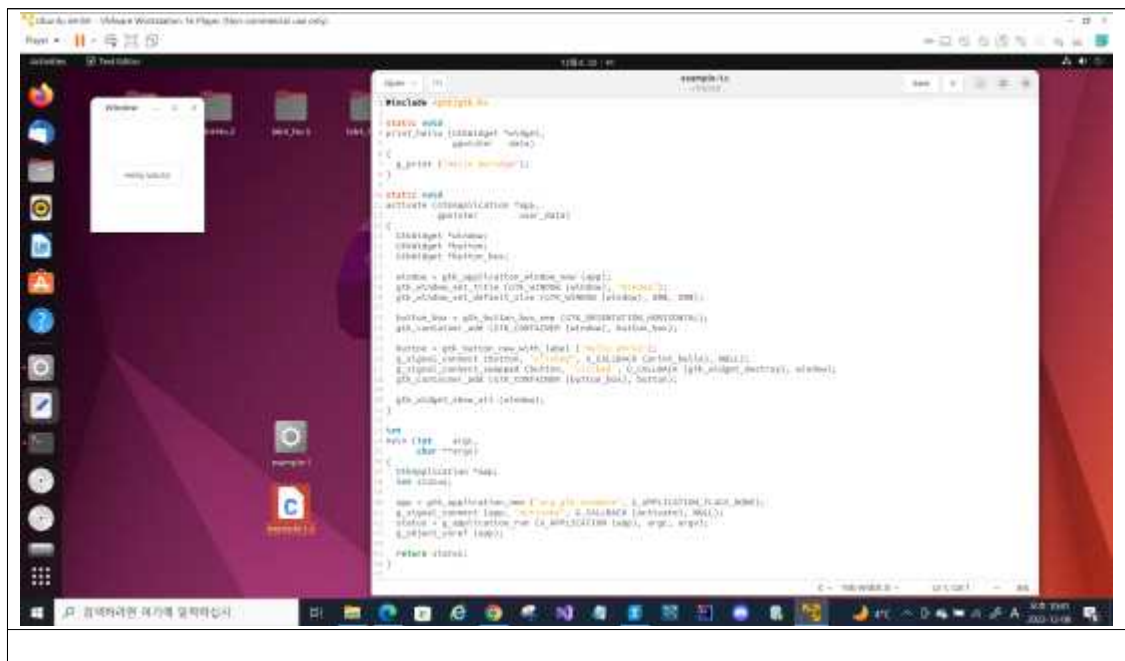
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <errno.h>
4 #include <string.h>
5 #include <sys/types.h>
6 #include <netinet/in.h>
7 #include <sys/socket.h>
8 #include <arpa/inet.h>
9
10 #define MYPORT 3490
11 #define MAXBUF 100
12
13 main(int argc, char* argv[]) {
14     /* 클라이언트 소켓 정보 획득 */
15     int csock, numbytes;
16     struct sockaddr_in serv_addr;
17     char buf[MAXBUF];
18     int len;
19
20     if(argc != 3){
21         fprintf(stderr, "Usage : tcp_client <SERVER IP> <ECHO STRING> \n");
22         exit(1);
23     }
24
25     if ((csock = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
26         perror("socket");
27         exit(1);
28     }
29
30     memset(&serv_addr, 0, sizeof(serv_addr));
31     serv_addr.sin_family = AF_INET;
32     serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
33     serv_addr.sin_port = htons(MYPORT);
34
35     if(connect(csock, (struct sockaddr *)&serv_addr, sizeof(struct sockaddr)) == -1){
36         perror("connect");
37         exit(1);
38     }
39
40     memset(buf, 0, MAXBUF);
41     strcpy(buf, argv[2]);
42     len = strlen(buf);
43
44     if (send(csock, buf, len, 0) != len) {
45         fprintf(stderr, "send failed...\n");
46         exit(1);
47     }
48 }

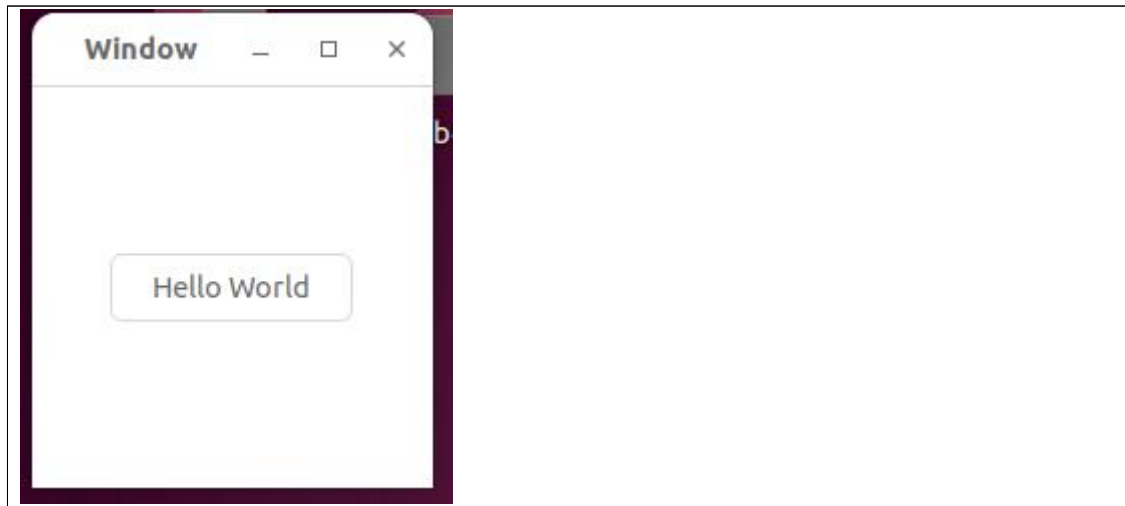
```

전체



8. GUI 관련 함수들을 사용하여 프로그램을 작성하고 실행하여 보고, 익숙해지도록 사용해 본다.





9. GTK+ 또는 Qt 를 이용하여 간단한 계산기 프로그램을 작성하여 본다.