|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **과 목** | 2020년 1학기 네트워크 프로그래밍 | | | **주 차** | 최종 |
| **분 류** | 텀 프로젝트 |
| **학 과** | 전자통신공학과 | **학 번** | 2015707003 | **성 명** | 정승훈 |
| **주 제** | Linux 환경 멀티 채팅 서버 및 클라이언트 프로그램 개발 | | | | |
| **코딩**  **조건** | 1. 멀티 채팅 서버 및 클라이언트 프로그램을 작성한다. 2. 서버 구현 방법   - 실행파일명은 ‘chat\_server\_process’, ‘chat\_server\_thread’, ‘chat\_server\_select’로 할 것.  - 포트번호는 3500번으로 고정 할 것.  - Multi-Processing, Multi-Threading, Select 방법 중 **1개 이상**을 선택하여 구현. 1개 이상인 경우 **약간의 가산점** 있어요.  - 클라이언트 접속 시 해당 클라이언트에 환영 메시지 출력하고 기존 접속되어 있는 클라이언트에게 접속 메시지를 전송 할 것.  - 다수의 클라이언트 접속을 처리하고 클라이언트가 보내오는 메시지를 다른 클라이언트 에게도 전송  - 클라이언트가 보내오는 시스템 메시지를 처리 할 것. (@show, @exit)  - 시스템 메시지 ‘@show’는 해당 클라이언트에게 현재 접속된 클라이언트 수와 아이디 명을 출력  - 시스템 메시지 ‘@exit’는 해당 클라이언트와 접속을 종료 할 것.  3. 클라이언트 구현방법  - 실행 파일명은 ‘chat\_client’로 할 것.  - 서버 아이피와 채팅방 아이디는 인수로 전송할 것.  - 실행 방법 : chat\_client 127.0.0.1 통돌이  - 본인이 원하는 방법으로 구현 할 것  4. 동작 예시  $ chat\_client 127.0.0.1 통돌이  통돌이님 멀티 채팅방에 오신 것을 환영합니다!  통순이님이 대화방에 입장하셨습니다.  통통이님이 대화방에 입장하셨습니다.  통통이 : 텀프로젝트다했니?  통순이 : 아직 하는중.  통통이 : 나 좀 갈켜줘~  통순이 : 안어려줘 금방해~  > 어렵던데 그러지말고 나두 좀 갈켜줘~ ← 입력메시지  통순이 : 알았어 다모여~ 그대신 술사~  > 넘넘 고마워~ ← 입력메시지  통통이 : 나도 고마워~  통통이님이 대화방을 나가셨습니다.  > @show  현재 서버에 접속중인 사용자는 아래와 같습니다.  통순이, 통돌이  > @exit  채팅이 종료되었습니다.  5. 소스코드와 스크린 캡처하여 리포트에 동작 화면 삽입 | | | | |

<source code>

1. chat\_client.c

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133 | /////////////////////////////////////////////////////////////////////////  // File Name       : chat\_client.c                                     //  // Date            : 2020/06/20 ~ 2020/06/30                           //  // OS              : Ubuntu 18.04.4 LTS                                //  // Student Name    : Seung Hoon Jeong                                  //  // Student ID      : 2015707003                                        //  // ------------------------------------------------------------------- //  // Title : Network Programming Term Project                            //  // Description : Linux 환경 멀티 채팅 서버 및 클라이언트 프로그램 개발 //  /////////////////////////////////////////////////////////////////////////  #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <string.h>  #include <sys/types.h>  #include <sys/socket.h>  #include <arpa/inet.h>  #include <fcntl.h>  #include <pthread.h>    #define BUFSIZE 1024  #define NAME 20    char name[NAME]; // each func use this variable    ///////////////////////////////////////////////////////////////////////  // Function : void \*send\_(void \*sock\_fd)                             //  // ================================================================= //  // Input: x                                                          //  // Output: x                                                         //  // Purpose: send message to server                                   //  ///////////////////////////////////////////////////////////////////////  void \*send\_(void \*sock\_fd)  {      int sock = \*(int \*)sock\_fd;      char named\_msg[BUFSIZE+NAME];      char msg[BUFSIZE];      while(1)      {          /\* 메세지 보내기전 메모리 초기화 필수 \*/          memset((void \*)msg, 0x00, BUFSIZE);          memset((void \*)named\_msg, 0x00, BUFSIZE+NAME);          read(STDIN\_FILENO, msg, BUFSIZE); // input message          /\* @exit 입력했을 때 \*/          if(!strcmp(msg, "@exit\n")) {              printf("채팅이 종료되었습니다.\n");              close(sock);              exit(0);            }          /\* @show 입력했을 때 \*/          else if(!strcmp(msg,"@show\n")) {              write(sock,msg,strlen(msg));          }            /\*그 이외 입력했을 때 \*/          else {              sprintf(named\_msg,"%s: %s", name, msg);              write(sock, named\_msg, strlen(named\_msg)); // 형식 맞춰서 서버로 전송          }      }      exit(0);  }      ///////////////////////////////////////////////////////////////////////  // Function : void \*recv\_(void \*sock\_fd)                             //  // ================================================================= //  // Input: x                                                          //  // Output: msg (talk or name\_list or exit\_msg)                      //  // Purpose: receive message from server                              //  ///////////////////////////////////////////////////////////////////////  void \*recv\_(void \*sock\_fd)  {      int sock = \*(int \*)sock\_fd;      char msg[BUFSIZE];      while(1)      {          memset((void \*)msg, '\0', sizeof(msg));          recv(sock, msg, BUFSIZE, 0); //서버로 온 메시지를 수신하기 위한 recv          printf("%s", msg); // 수신받은 메시지 출력      }      exit(0);  }    /////////////////////////////////////////////////////////////////////////////////////////////////////  // Function : int main(int argc, char \*\*argv)                                                      //  // ==============================================================================================  //  // Input: ./[실행파일] [IP] [USERNAME]                                                             //  // Output: x                                                                                      //  // Purpose: send message to server and receive message from server                                //  ////////////////////////////////////////////////////////////////////////////////////////////////////  int main(int argc, char\* argv[])  {      int sockfd;      char buf[BUFSIZE];      pthread\_t send\_thread, recv\_thread;      struct sockaddr\_in serv\_addr;      /\* check argument count \*/      if(argc != 3){          printf("Usage : %s <IP> <name>\n", argv[0]);          exit(1);      }      //인자로 받은 name을 sprintf 함수를 써서 name에 넣어준다.      sprintf(name, "%s", argv[2]);        /\* open socket \*/      sockfd = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP);        /\*initialize server socket information struct to zero and add info\*/      memset((void \*)&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(3500);        /\*connect to server \*/      if(connect(sockfd, (struct sockaddr\*)&serv\_addr, sizeof(serv\_addr))==-1)          perror("connect");        /\* if connect successful, send name info to server \*/      write(sockfd, name, sizeof(name));        /\* sending thread, receiving thread production \*/      pthread\_create(&send\_thread, NULL, send\_, &sockfd);      pthread\_create(&recv\_thread, NULL, recv\_, &sockfd);        /\*call join func for successful resource return \*/      pthread\_join(send\_thread, NULL);      pthread\_join(recv\_thread, NULL);        close(sockfd);      return 0;  }  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

1. chat\_server\_select.c

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171 | /////////////////////////////////////////////////////////////////////////  // File Name       : chat\_server\_select.c                              //  // Date            : 2020/06/20 ~ 2020/06/30                           //  // OS              : Ubuntu 18.04.4 LTS                                //  // Student Name    : Seung Hoon Jeong                                  //  // Student ID      : 2015707003                                        //  // ------------------------------------------------------------------- //  // Title : Network Programming Term Project                            //  // Description : Linux 환경 멀티 채팅 서버 및 클라이언트 프로그램 개발 //  /////////////////////////////////////////////////////////////////////////  #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <sys/types.h>  #include <sys/socket.h>  #include <arpa/inet.h>  #include <unistd.h>    #define MAXUSER 1000  #define MAXLINE 1024  #define NAME 20  #define PORTNUM 3500  #define LISTENQ 20    int main ()  {      int listenfd, connfd;      int clilen, cli\_num = 0;      int maxfd;      int fdnum;      char \*ptr; //골뱅이 문자 위치 찾기 위한 포인터      char welcome\_msg[MAXLINE];      char exit\_msg[MAXLINE];      int client[MAXUSER] = {0};      char name[MAXUSER][NAME];      char buf[MAXLINE];      char buff[MAXLINE];      char show\_list[MAXLINE];      char login\_state[MAXLINE];      struct sockaddr\_in server\_addr, client\_addr;      fd\_set fds\_read, allfds;        /\* open socket \*/      if((listenfd = socket(AF\_INET, SOCK\_STREAM, IPPROTO\_TCP)) < 0) {          perror("socket");        }      /\* prevent bind error after server terminated \*/      int option = 1;      if(setsockopt(listenfd, SOL\_SOCKET, SO\_REUSEADDR, &option, sizeof(option)) < 0) {          perror("setsockopt");          exit(0);      }        /\*initialize server socket information struct to zero and add info\*/      memset((void\*)&server\_addr, 0x00, sizeof(server\_addr));      server\_addr.sin\_family = AF\_INET;      server\_addr.sin\_addr.s\_addr = htonl(INADDR\_ANY);      server\_addr.sin\_port = htons(PORTNUM);        /\* bind socket \*/      if(bind(listenfd, (struct sockaddr \*)&server\_addr, sizeof(server\_addr)) < 0) {          perror("bind");          exit(0);      }        /\* listen queue production \*/      if((listen(listenfd, LISTENQ)) == -1) {          perror("listen");          exit(0);      }        FD\_ZERO(&fds\_read); //fd\_set initialization      FD\_SET(listenfd, &fds\_read); //listenfd를 fd\_set에 setting      maxfd = listenfd; //send max value to select()        while(1) {          allfds = fds\_read; // copy본 store (select 사용전에)          /\*select함수 인자 순서대로,          fd 값중 max value, read event check, write check, execpt check, timeout value. NULL infinite wait\*/          fdnum = select(maxfd + 1, &allfds, (fd\_set \*)0, (fd\_set \*)0, NULL);          //when an read event occurs to listenfd , Please accept          if(FD\_ISSET(listenfd, &allfds)) {              clilen = sizeof(client\_addr);              connfd = accept(listenfd, (struct sockaddr \*)&client\_addr, &clilen);              FD\_SET(connfd, &fds\_read); //accepted socket fd register              client[connfd] = connfd; //login client storing                read(connfd, name[connfd], NAME); // receive name sending client to server                strcpy(buff,name[connfd]); //send message to only request client              strcat(buff,"님 멀티 채팅방에 오신 것을 환영합니다.\n");              cli\_num++; // login user num check              write(client[connfd],buff, strlen(buff));              bzero(buff,sizeof(buff));              //printf("check name : %s\n", name[accept\_sock]);              if(connfd > maxfd) // if accepted socketfd가 기존의 max값보다 크다면                  maxfd = connfd; // exchange                sprintf(welcome\_msg, "%s님이 채팅방에 입장했습니다.\n", name[connfd]);                /\*send message to rest process(login state) \*/              for(int i = 0; i <= maxfd; i++) {                  if(client[i] != client[connfd])                      send(client[i], welcome\_msg, sizeof(welcome\_msg), 0);              }                //printf("[%d] Accept OK\n", accept\_sock); // accept check              continue;          }            /\* when No accept request \*/          for (int check\_fd = 0; check\_fd <= maxfd; check\_fd++) { // loop searching, find to applicable socket fd              //어떤 fd 값이 select read 이벤트 검사한 곳에서 이벤트 요청한게 확인 됐다면              if (FD\_ISSET(check\_fd, &allfds)) {                  bzero(buf,sizeof(buf));                  if((read(check\_fd, buf, MAXLINE)) <= 0) {                      /\* when other user disconnected \*/                      close(check\_fd);                      FD\_CLR(check\_fd, &fds\_read);                      //if(client\_array[check\_fd] != client\_array[accept\_sock])                      //sprintf(exit\_msg,"%s님이 대화방을 나갔습니다.\n", name[check\_fd]);                      strcpy(exit\_msg,name[check\_fd]);                      strcat(exit\_msg,"님이 대화방을 나갔습니다.\n"); // exit\_msg production                      cli\_num--; // login user num check                      for(int j=0; j<=maxfd; j++)                          if(client[j] == j)                              send(client[j],exit\_msg,sizeof(exit\_msg),0); // send exit\_msg to all login state client (except own client)                      bzero(exit\_msg,sizeof(exit\_msg));                      client[check\_fd] = 0;                      memset((void \*)name[check\_fd], 0x00, sizeof(name[check\_fd])); // initialize client's fd and name                  }              else {                  ptr = strchr(buf, '@'); //if sending message includes '@'가 있다면 return value to ptr applicable location                  if (ptr != NULL)  {                      if(!strcmp(ptr, "@show\n")) { //is msg @show? check                          bzero(show\_list,sizeof(show\_list));                          strcpy(show\_list,"현재 접속중인 사용자는 다음과 같습니다.\n");                          if(client[check\_fd] == check\_fd);//check fd                          for (int i = 0; i <= maxfd; i++) { //send name to all login state process                              if(name[i][0] != '\0') {                                  strcat(show\_list,name[i]);                                  strcat(show\_list," ");                                  //strcat(show\_list,"현재 접속자 수:");                                  //strcat(show\_list,cli\_num);                              }                          }                          //bzero(login\_state,sizeof(login\_state));                          sprintf(login\_state,"현재 접속자 수:%d\n",cli\_num);                          strcat(show\_list,login\_state);                          write(client[check\_fd], show\_list, sizeof(show\_list));                          }                  }                    //printf("Read : %s" ,buf); //server check ex                  for(int i = 0; i <= maxfd; i++) {                      if(client[i] != 0)                          if(strcmp(buf,"@show\n"))                              send(client[i], buf, sizeof(buf), 0); //send message to all login state client(except own client)                      }                  }                  if(--fdnum <= 0) break;              }          }      }      close(listenfd);      return 0;  }    [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

<Result>

