프로젝트 계획서

유닉스 마피아 게임

201511041 이동석

201411079 강대훈

- 목 차 -

**1. 요구사항 분석**

**2. 프로젝트 계획**

2.1 프로젝트 계획

2.2 산출물 관리

**3. 결과**

3.1 프로그램 구조

3.2 프로그램 사용 설명

3.3 파일 설명

3.4 소스 코드

**4. 테스트 결과**

**5. 추후 발전 방향**

# 요구사항 분석

저희 팀 이팔청춘은 유닉스에서 배운 내용을 토대로 마피아 게임을 만들 것입니다.

게임 설명 : 게임에 직접 참여하지 않는 사회자 1명은 반드시 필요하다. 소수의 마피아, 다수의 일반시민으로 나뉘어 서로의 생존을 가리는 게임이다. 낮과 밤으로 시간이 나뉘어 있으며 낮에는 모두가 참여하는 인민재판투표로, 밤에는 마피아들만의 비밀회의로 죽을 사람을 결정한다. 일반시민은 모든 마피아를 죽여야 승리하며, 마피아는 자기들의 수와 일반시민의 수가 같아지면 승리한다.

1. 사회자는 서버 게임 참여자는 클라이언트

* 서버가 마피아 게임에서의 진행자 역할, 클라이언트가 게임을 참여합니다.

1. 참여자가 5명 이하일 때는 마피아 1명, 5명보다 많을 때는 마피아 2명으로 게임을 구현

* 참여자의 수에 따라서 임의의 클라이언트에게 마피아를 지정해 주었습니다

1. 서버와 클라이언트 간의 통신은 메시지 큐를 이용해 구현

- 특정한 타입을 기다리거나 보냄으로써 사회자와 플레이어 간의 통신을 구현했습니다.

1. 모든 클라이언트는 공유 메모리를 통해 대화

* 공유 메모리에 자신이 입력한 문자열을 저장해 언제든지 볼 수 있게 했습니다.

1. 마피아 간의 대화는 소켓을 이용

* 소켓을 기반으로 한 채팅 프로그램을 만들어 마피아 간의 대화가 가능하게 합니다.

1. 참여자가 자신의 프로세스 번호를 알게 했음

* getpid 함수를 사용해서 참여자가 자신의 프로세스 번호를 알 수 있게 했습니다.

1. 참여자의 생사 등 게임의 기록은 파일 입출력을 이용해 관리

* txt파일을 fopen 함수를 사용해 만들어 전적 등 게임의 기록들을 관리했습니다

# 프로젝트 계획

## 주요 일정

- 주요 일정

|  |  |  |
| --- | --- | --- |
| 단계 | 일정 | 산출물 |
| 요구사항 분석 | 2019-11-25 ~ 2019-11-25 | 요구사항 명세서 |
| 프로젝트 계획 | 2019-11-25 ~ 2019-11-25 | 프로젝트 계획서 |
| 설계 | 2019-11-25 ~ 2019-11-27 | 설계서 |
| 구현 | 2019-11-27 ~ 2019-12-03 | 소스코드 |
| 테스팅 | 2019-12-03 ~ 2019-12-03 | 테스트케이스 , 보고서 |
| 유지보수 | 2019-12-03 ~ 2019-12-09 |  |

## 조직 및 역할 분담

- 역할

|  |  |  |
| --- | --- | --- |
| 팀원 | 역할 | 책임 |
| 이동석 | PM 및 프로그램 설계 | 프로젝트의 계획과 실행의 종합적인 책임 |
| 강대훈 | 프로그램 설계 및 문서 작성 | 요구사항에 맞춰서 프로그램을 설계하고 문서를 작성 |

## 산출물 관리

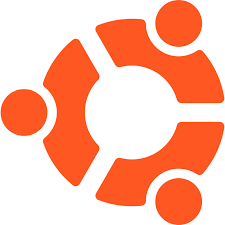
Trello

문서관리

GitHub

코드관리

- 도구

 Ubuntu : 리눅스 커널을 기반으로 한 리눅스 배포판

- 개발 환경

Git (Github) : 분산 버전 관리 시스템 (원격 저장소)

- 소스코드의 버전관리 (전체 이력을 기록)

 Trello : 이슈 트래킹 시스템

- 요구사항, 문서, 회의 내용 등을 등록 및 관리

파일을 올리면 팀원들의 feedback, 수정 후 재 업로드 형식

- Github를 이용한 프로그램 코드 관리

- Trello를 이용한 문서 관리

# 결과

## 프로그램 구조

서버와 클라이언트 간의 통신은 메시지 큐를 사용. 투표를 하기 전 토론은 공유 메모리를 이용해 사용했고, 마피아 간의 통신은 소켓을 이용해 구현했다.

: 클라이언트가 서버에게 보내는 타입 1의 메시지

: 서버가 클라이언트의 메시지 타입을 알려주기 위한 타입 2의 메시지

: 소켓으로 마피아 간의 통신

: 공유 메모리를 이용한 클라이언트 간의 통신

## 프로그램 사용 설명

- 서버는 프로그램으로 구현한 사회자 역할이므로 실제 게임은 클라이언트끼리 이루어진다.

먼저 컴파일 된 서버를 실행시키고 서버에서 참여할 클라이언트의 숫자를 입력하면 그 수만큼 클라이언트를 실행하면 서버는 랜덤으로 마피아를 지정하고 게임을 시작한다.

.

## 파일 설명

- mg\_server : 사회자 역할을 맡은 서버 파일.

- mg\_client : 게임 플레이어 역할을 맡은 클라이언트 파일

## 소스 코드

- mg\_server.c

|  |
| --- |
| 헤더 파일 추가 및 함수 선언 |
| 1 #include "mg\_header.h"  2  3 int client\_num = 4; // 기본 값.  4 int discussion\_time = 10; // 기본 10초.  5 int killnum = -1; // 죽는 자의 번호.  6  7 struct mymsgbuf mesg;  8 int msgid, len;  9 key\_t key;  10 key\_t sh\_key;  11 // 메세지 관련 변수들.  12  13 struct sigaction act;  14 // sigaction 구조체 포인터 변수.  15  16 FILE \*rfp, \*wfp;  17 // 파일 구조체 포인터  18  19 char \* nameofjobs[5]={"host","mafia","citizen",};  20 // 직업 명 문자열을 가리키는 포인터 배열.  21  22 int nopb\_jobs[5] = {0,}; // 직업별 사람수.  23 //0: host, 1: mafia, 2 : citizen ...  24  25 int p\_pids[MAX\_PEOPLE] ; // people pids.  26 char \* p\_nics[MAX\_PEOPLE]={0,}; // peple nicknames  27 int p\_jobs[MAX\_PEOPLE] ; // people jobs  28 // 죽은 면 -1 로 저장하자.  29  30 int b\_p\_jobs[MAX\_PEOPLE]; // people jobs 의 backup 본.  31  32 void print\_start\_message();  33 void receive\_print\_message(int mtype);  34 void send\_message(int mtype, char \* message);  35 void random\_job\_make(int client\_num, int arr[MAX\_PEOPLE]);  36 void test\_job\_make();  37 void send\_receive\_first\_message();  38  39 void write\_user\_data(); // 데이터를 실제로 쓰는 부분.  40 void save\_user\_data\_send\_messages();  41 void save\_final\_data();  42  43  44 void print\_kill\_one\_check\_condition();  45 int check\_end\_condition();  46 // void print\_kill~ 에서 쓰이는 함수. 종료 조건에 도달하면 1을 반환. 아니면  47  48 void become\_night();  49 void become\_day();  50  51   1. // 함수의 선언부. |
| 1. :mg\_header.h 헤더   2 ~ 30 : 각 함수에 필요한 전역 변수 및 포인터 선언   1. ~ 49 : 함수의 선언 |

|  |
| --- |
| void print\_start\_message() |
| 55 void print\_start\_message(){  56 printf("안녕하세요 마피아 게임입니다.\n");  57 printf("당신은 게임의 사회자로 선정되었습니다.\n");  58 printf("제작 자 : 강대훈, 이동석.\n");  59 printf("접속 할 사람수는 몇명입니까?\n");  60 printf("입력 : ");  61 } |
| 게임을 시작할 때 호출하는 함수. |

|  |
| --- |
| **-** void receive\_print\_message(int mtype), void send\_message(int mtype, char \* message) |
| 63 void receive\_print\_message(int mtype){  64 len = msgrcv(msgid, &mesg, 800, mtype, 0);  65 printf("Received Msg = %s \n", mesg.mtext);  66 }  67  68 void send\_message(int mtype, char \* message){  69 mesg.mtype = mtype;  70 strcpy(mesg.mtext, message);  71 if (msgsnd(msgid, (void \*)&mesg, 800, IPC\_NOWAIT) == -1) {  72 perror("msgsnd");  73 exit(1);  74 }  75 } |
| void receive\_print\_message(int mtype) : 특정한 메시지를 기다리고, 그 내용을 출력하는 함수.  void send\_message(int mtype, char \* message) : 각각의 매개변수를 메시지 구조체에 채우고 메시지를 보내는 함수.  63 ~ 66 : mtype의 메시지 타입을 기다리고 그 타입의 메시지 내용을 출력한다.  68 ~ 75 : mtype, message를 각각 메시지에 채워놓고 메시지를 보낸다. |

|  |
| --- |
| - random\_job\_make(int client\_num, int arr[MAX\_PEOPLE]) |
| 78 void random\_job\_make(int client\_num, int arr[MAX\_PEOPLE]){  79 for(int i=0;i<client\_num;i++)  80 {  81 arr[i]=2;  82 }  83  84 int cnt = 0;  85 int temp=0;  86 srand(time(NULL));  87  88 if(client\_num<=5)  89 {  90 int r = rand()%client\_num;  91 arr[r]=1;  92  93 }  94 else if(client\_num>5)  95 {  96  97 int r = rand()%client\_num;  98 arr[r]=1;  99 temp = r;  100 cnt++;  101 while(1)  102 {  103 r = rand()%client\_num;  104 if(cnt==2)  105 break;  106 if(temp == r)  107 {  108 continue;  109 }  110 arr[r]=1;  111 temp = r;  112 cnt++;  113 }  114 }  115 } |
| - random\_job\_make(int client\_num, int arr[MAX\_PEOPLE]) : 일정한 조건에 따라 마피아를 랜덤으로 배정해주는 함수.  79 ~ 86 : 일단 모든 클라이언트의 직업을 시민(2)로 배정하고 조건문에 사용할 변수를 생성한다.  88 ~ 93 : 클라이언트의 수가 5명 이하일 경우 랜덤으로 마피아 한 명을 배정한다  94 ~ 105 : 클라이언트의 수가 5명보다 클 경우 마피아를 두 명에게 배정해준다.  여기서 마피아가 배정이 되면 cnt변수의 값을 증가시키고 값이 2가 될 시 while문을 종료한다. |

|  |
| --- |
| void send\_receive\_first\_message() |
| 126 void send\_receive\_first\_message(){  127 key = ftok("mg", 1); // mg means "mafia game".  128 // key를 만듭니다.  129 if ((msgid = msgget(key, IPC\_CREAT|0644)) < 0) {  130 perror("msgget");  131 exit(1);  132 }  133 // 해당 키 값으로 message 식별자를 만듭니다.  134 // 사람들 직업번호가 저장된 배열.  135 nopb\_jobs[Host] = 1;  136 // 사회자는 항상 하나  137  138 if(client\_num<=5){  139 nopb\_jobs[Mafia] = 1;  140 nopb\_jobs[Citizen] = client\_num -1;  141 }  142 if( client\_num>=6){  143 nopb\_jobs[Mafia] =2;  144 nopb\_jobs[Citizen] = client\_num -2;  145 }  146 //인원수에 따라 마피아, 시민수를 조절하는 과정입니다.  147  148 //test\_job\_make()  149 random\_job\_make(client\_num, p\_jobs);  150 // 클라이언트의 수에 따라 랜덤하게 마피아와 시민을 정하는 코드 입니다.  151  152 for(int i=0;i<client\_num;i++){  153 b\_p\_jobs[i] =p\_jobs[i];  154 // 백업 해두기.  155 }  156 // b\_p\_jobs 에 p\_jobs 에 있던 값을 저장합니다.  157 // 그 이유는 특정 i 번호에 해당하는 프로세스가 죽으면, 해당 사람의 직업이 -1이 되기 때문입니다.  158  159 for(int i=0;i<MAX\_PEOPLE;i++){  160 p\_nics[i] = malloc(sizeof(char) \* 50);  161 }  162 // 닉네임을 저장할 공간을 할당하는 과정.  163  164 int mafia\_flag = 0 ; // 처음은 클라이언트.  165  166 for(int i=0;i<client\_num;i++){  167  168 receive\_print\_message(1);  169 char \* strptr = strtok(mesg.mtext, "-");  170 strcpy(p\_nics[i],strptr);  171 strptr = strtok(NULL, "-");  172 p\_pids[i] = atoi(strptr);  173 // 해당 메세지를 받아, 거기서 nicsname과 pid를 추출하는 과정.  174  175 memset(mesg.mtext, 0, sizeof(mesg.mtext));  176 sprintf(mesg.mtext, "%d", i+3);  177 send\_message(2, mesg.mtext);  178 // 참여자에게 관련 참여자가 사용할 mtype 번호를 메세지큐에 보냅니다.  179 // 이때는 타입 2번 메세지를 사용하여 보내고  180 //참여자들은 3번부터 mtype을 부여받게 됩니다.  181  182 memset(mesg.mtext, 0, sizeof(mesg.mtext));  183 sprintf(mesg.mtext, "%d", p\_jobs[i]);  184 send\_message(i+3, mesg.mtext);  185 // mtype i+3 번으로 직업번호를 보내주자.  186 // 해당 프로세스의 mtype으로 직업번호를 보냅니다.  187  188  189 if(p\_jobs[i] == Mafia){  190 memset(mesg.mtext, 0, sizeof(mesg.mtext));  191 sprintf(mesg.mtext, "%d", nopb\_jobs[Mafia]);  192 send\_message(i+3, mesg.mtext);  193 // 보낸 프로세스의 직업이 마피아라면  194 // 마피아에게 보낼 첫번째 메세지 : 마피아 수.  195  196 if(nopb\_jobs[Mafia] >=2){  197 // 처음에 보내는 것은 당신이 mafia\_client인가? mafia\_server인가?  198 // mafia\_client면 0, mafia\_server면 1.  199 memset(mesg.mtext, 0, sizeof(mesg.mtext));  200 sprintf(mesg.mtext, "%d", mafia\_flag);  201 send\_message(i+3, mesg.mtext);  202 // 처음은 0보내고, 클라이언트.  203 //두번쨰 녀석은 1. 서버  204 // 마피아에게 보낼 두번째 메세지 : 마피아 서버인가 클라이언트 인가?.  205  206 //그다음 메세지.  207 mafia\_flag =1;  208 char mafia\_info[500] = {0,};  209 strcpy(mafia\_info, "마피아 정보입니다.\n");  210 strcat(mafia\_info, "처음에만 알려주니 잘 기억하세요.\n");  211 strcat(mafia\_info, "번호\tnickname\n");  212 for(int j=0;j<client\_num;j++){  213 if(p\_jobs[j] == Mafia){  214 char numstr[10] = {0,};  215 sprintf(numstr, "%d", j);  216 strcat(mafia\_info, numstr);  217 strcat(mafia\_info,"\t");  218 strcat(mafia\_info, p\_nics[j]);  219 strcat(mafia\_info, "\n");  220 }  221 }  222  223 send\_message(i+3, mafia\_info);  224 // 마피아에게 보낼 세번째 메세지 : 마피아 정보.  225 }  226  227 // 여기서 마피아가 두명 이상일 때 어떻게 처리할 지 고민해야 할 것이다.  228 }  229 }  230 } |
| void send\_receive\_first\_message() : 인원 수에 따라 마피아의 수를 조절하고 마피아가 2명일 시 소켓 통신을 위해 마피아에게 서버, 클라이언트를 지정해주는 함수.  127 : mg라는 키 생성  129 ~ 134: 메시지 큐 생성. 메시지 큐가 만들어져 있지 않으면 644권한으로 생성헌다.  135 : nopb\_jobs[] 라는 배열은 직업의 수를 저장하는 배열. 사회자는 반드시 1명만 있어야 한다.  138~ 140 : 플레이어의 수가 5명 이하일 때 nopb\_jobs[Mafia]에 1을 저장하고, nopb\_jobs[Citizen]에 플레이어의 수에서 마피아 수를 뺀 숫자만큼 저장한다. 마찬가지로 플레이어의 수가 6명일 때는 nobp\_jobs[Mafia]에 2를 저장하고 nopb\_job[Citizen]에는 플레이어의 수 – 2 를 저장한다.  149 : random\_job\_make(client\_num, p\_jobs) 함수로 랜덤하게 마피아와 시민을 지정함  152 ~ 154 : 사용자가 죽임을 당할 시 p\_jobs[]의 값을 -1로 지정하므로 게임이 끝나고 플레이어에게 직업을 알려주기 위해서 b\_p\_jobs[]에 백업을 해놓는다.  159 ~ 160 : 닉네임을 저장할 공간을 할당 받는다.  164 : 마피아 플래그이다. 플래그가 0일시 클라이언트, 1일시 서버 역할  168 : 클라이언트로부터 타입이 1인 메시지를 기다린다. 클라이언트는 자신의 닉네임과 pid를 타입 1인 메시지로 서버에게 보낸다.  169 ~ 174 : “-“를 기준으로 문자열을 나누고 닉네임과 pid를 추출하는 과정  175~ 178 : 메시지 타입이 2이고 클라이언트가 사용할 메시지의 타입 (3부터) 메시지 타입을 내용으로 담아 각각의 클라이언트에게 보낸다  182 ~ 184 : 그리고 클라이언트의 메시지 타입 (3부터)으로 직업 번호를 보낸다.  189 ~ 201 : 만약 프로세스의 직업이 마피아면 마피아에게 마피아의 수를 보내는데 마피아의 수가 2이면 추가로 클라이언트 마피아, 서버 마피아인지 알려준다.  207~ 223 : 마피아들에게 마피아의 인덱스 번호와, 닉네임을 보낸다. Strcat 함수로 mafia\_info에 붙여서 한번에 보낸다. |

|  |
| --- |
| void write\_user\_data() |
| 232 void write\_user\_data(){  233 // 사용자들에게 공지할 자료 입니다.  234 // process에 할당된 index 번호, nickname, pid , 살아있는 지 유무를 알수 있습니다.  235 // 그런한 자료를 만드는 과정.  236  237 printf("\n\nuser\_data.txt 갱신을 시작합니다.\n");  238 if ((wfp = fopen("user\_data.txt", "w")) == NULL) {  239 perror("fopen: user\_data.txt");  240 exit(1);  241 }  242 fprintf(wfp,"-------user\_data\_file------\n");  243 fprintf(wfp,"--num nic pid -----\n");  244 for(int i=0;i<client\_num;i++){  245 if(p\_jobs[i] == -1){  246 fprintf(wfp,"%d : %s %d DEAD\n", i, p\_nics[i], p\_pids[i]);  247 }  248 else {  249 fprintf(wfp,"%d : %s %d ALIVE\n", i, p\_nics[i], p\_pids[i]);  250 }  251 }  252 fclose(wfp);  253  254 printf("user\_data.txt 갱신되었습니다!\n");  255 } |
| void write\_user\_data() : 사용자들의 프로필을 파일 입출력 형식으로 저장하고 투표로 사람이 죽거나 마피아가 시민을 죽였을 때 사용자들의 생사 여부를 저장하는 함수.  238 : fopen으로 user\_date.txt 파일을 쓰기 권한으로 열음  242 ~ 252 : fprintf안에 있는 정보들을 user\_data.txt에 사용자 인덱스, 닉네임, 프로세스 번호를 저장한다. 여기서 p\_jobs[i]의 값이 -1이면 DEAD, -1이 아니면 ALIVE라는 문자열을 뒤에 쓴다. |

|  |
| --- |
| void save\_user\_data\_send\_messages() |
| 257 void save\_user\_data\_send\_messages(){  258 write\_user\_data();  259 // 사용자들에게 공지할 자료 입니다.  260 // process에 할당된 index 번호, nickname, pid , 살아있는 지 유무를 알수 있습니다.  261 // 그런한 자료를 만드는 과정.  262  263 for(int i=0;i<client\_num;i++){  264 if(p\_jobs[i] == -1){  265 continue;  266 }  267 send\_message(i+3, "User\_data is sent by host");  268 }  269 // 자료가 만들어졌다고 알려주는 과정.  270 } |
| void save\_user\_data\_send\_messages(): write\_user\_data()를 호출하고 send\_message()로 user\_data.txt의 데이터를 보내는데 p\_jobs[i]의 값이 -1일 때는 죽었다는 표시이므로 보내지 않는다. |

|  |
| --- |
| void save\_final\_data() |
| 272 void save\_final\_data(){  273 printf("save final data is called\n");  274 if ((wfp = fopen("user\_data.txt", "w")) == NULL) {  275 perror("fopen: user\_data.txt");  276 exit(1);  277 }  278 fprintf(wfp,"-------user\_data\_file------\n");  279 fprintf(wfp,"--num nic pid job-----\n");  280 for(int i=0;i<client\_num;i++)  281 {  282 if(p\_jobs[i] == -1){  283 fprintf(wfp,"%d : %s %d %s DEAD\n", i, p\_nics[i], p\_pids[i], nameofjobs[b\_p\_jobs[i]]);  284 }  285 else {  286 fprintf(wfp,"%d : %s %d %s ALIVE\n", i, p\_nics[i], p\_pids[i], nameofjobs[b\_p\_jobs[i]]);  287 }  288 }  289  290 fclose(wfp);  291 } |
| void save\_final\_data() : 게임이 끝나고 플레이어의 최종 상태를 user\_data.txt 파일에 저장. 플레이어의 직업도 추가로 저장한다. |

|  |
| --- |
| int check\_end\_condition() |
| 293 int check\_end\_condition(){  294 if(nopb\_jobs[Citizen]<=nopb\_jobs[Mafia]){  295 save\_final\_data();  296 // 게임이 끝나서 직업 까지 알수 있는 정보를 게시.  297 // 마피아 승리 조건.  298 for(int i=0;i<client\_num;i++){  299 if(p\_jobs[i] == -1){  300 continue;  301 }  302 kill((pid\_t)p\_pids[i], SIGUSR1);  303 }  304 // 모든 client에게 SIGUSR1을 보냄.  305 // 마피아의 승리를 의미.  306 return 1;  307 }  308 else if(nopb\_jobs[Mafia] <=0){  309 save\_final\_data();  310 // 시민승리 조건.  311 for(int i=0;i<client\_num;i++){  312 if(p\_jobs[i] == -1){  313 continue;  314 }  315 kill((pid\_t)p\_pids[i], SIGUSR2);  316 // 모든 client에게 SIGUSR2을 보냄.  317 // 시민의 승리를 의미.  318 }  319  320 // 모든 client에게 signal을 보냄.  321 return 1;  322 }  323 else  324 return 0;  325 } |
| int check\_end\_condition() : 게임의 종료조건을 체크하는 함수.  294 ~303 : 마피아의 승리 조건 (시민의 수 <= 마피아 수 )일 때, SIGUSR1라는 시그널을 보낸다.  308~315 : 시민의 승리 조건 (모든 마피아 사망)일 때, SIGUSR2 라는 시그널을 보낸다.  321: 게임이 끝났을 시 1을 반환  323: 게임이 끝날 조건이 아닐 시 0을 반환 |

|  |
| --- |
| void print\_kill\_one\_check\_condition() |
| 327 void print\_kill\_one\_check\_condition(){  328 printf("죽은 사람은 %d 입니다.\n", killnum);  329 nopb\_jobs[p\_jobs[killnum]]--;  330 p\_jobs[killnum] = -1; // 해당 번호를 죽은 것으로 표시.  331  332 kill((pid\_t)p\_pids[killnum], SIGQUIT);  333 // 해당 pid로 시그널 보내기.  334  335 //check\_end\_condition  336 // 끝나는 조건에 도달했는지 확인.  337 if(check\_end\_condition() == 1){  338  339 // 자신을 종료..  340 exit(1);  341 }  342 } |
| void print\_kill\_one\_check\_condition() : 죽은 사람을 알려주고, 그 사람의 직업의 수 하나를 줄인다. p\_job[killnum]을 -1로 표시해 죽은 것으로 표시하고 그 프로세스로 SIGQUIT이라는 시그널을 보낸다. 그리고 check\_end\_condition이 1일 때는 게임이 끝나는 조건이므로 exit(1)으로 프로그램을 종료한다. |

|  |
| --- |
| void become\_night() |
| 344 void become\_night(){  345 //2.1 모두에게 밤이 되었음을 알림.  346 printf("밤이 되었습니다.\n모두 엎드려주세요.\n마피아는 고개를 들어 서로를 확인하시고\n토의시간을 거쳐 죽일 사람을 선택하세요.\n시민은 잠시 기다려주세요.\n사회자는 마피아의 선택을 기다립니다.\n");  347 for(int i=0;i<client\_num;i++){  348 if(p\_jobs[i] == -1){  349 continue;  350 // 무시.. 죽었으니까.  351 }  352 send\_message(i+3, "밤이 되었습니다.\n모두 엎드려주세요.\n마피아는 고개를 들어 서로를 확인하시고\n토의시간을 거쳐 죽일 사람을 선택하세요.\n시민은 잠시 기다려주세요.\n");  353 }  354  355 //2.2 마피아가 신호를 보내주기를 기다림.  356 for(int i=0;i<nopb\_jobs[Mafia];i++){  357 receive\_print\_message(1);  358 }  359  360 //2.3 해당 넘버를 죽은 것으로 표시하고 종료 메세지를 보냄.  361 killnum = atoi(mesg.mtext);  362 // 여기서  363  364 print\_kill\_one\_check\_condition();  365 // 1명을 죽이고 종료 조건을 체크함.  366 for(int i=0;i<client\_num;i++){  367 if(p\_jobs[i] == -1){  368 continue;  369 // 무시.. 죽었으니까.  370 }  371 if(p\_jobs[i] == Mafia){  372 memset(mesg.mtext, 0, sizeof(mesg.mtext));  373 sprintf(mesg.mtext, "%d", nopb\_jobs[Mafia]);  374 send\_message(i+3, mesg.mtext);  375 }  376 // 보내는 메세지 받는 사람이 마피아면 현재 마피아 수를 추가적으로 보내준다.  377 }  378  379 write\_user\_data();  380 // 낮이 되기 전에 최신 정보를 저장함.  381  382 } |
| void become\_night() : 게임에서 밤을 구현한 함수.  345 ~ 352 : p\_jobs[i]가 -1인 값을 제외하고 send\_message() 밤이 되었다는 메시지를 보낸다  356 ~ 357 : 마피아가 사회자에게 메시지를 보내기를 기다린다.  361 ~ 364 : 마피아는 사용자 인덱스 번호를 text로 하는 메시지를 보내므로 atoi함수를 사용해 정수형태로 바꾸고, 그 정수에 해당되는 사용자를 죽인다.  366 ~ 377 : 마피아에게 현재 마피아의 수를 추가로 보내는 코드.  379 : 최신 데이터 업데이트 |

|  |
| --- |
| int vote\_result(int whowilldie[]) |
| 384 int vote\_result(int whowilldie[]){  385 int maxindex = 0;  386 int max = -1;  387 int samecnt =0;  388 for(int i=0;i<client\_num;i++){  389 if(p\_jobs[i] == -1){  390 continue;  391 }  392 if(max<whowilldie[i]){  393 samecnt = 0;  394 max = whowilldie[i];  395 maxindex =i;  396 }  397 else if(max == whowilldie[i]){  398 samecnt =1;  399 }  400 }  401 if(max <= 0){  402 return -1;  403 }  404 if(samecnt ==1){  405 return -1;  406 }  407  408 return maxindex;  409 } |
| int vote\_result(int whowilldie[]) : 투표 결과에 따른 리턴 값을 반환하는 함수  385 ~ 387 : 변수 선언  388 ~ 400 : 죽은 플레이어일 때는 그냥 넘기고 whowilldie[i]의 값이 max보다 클 경우 samecnt를 0으로 지정하고 max를 whowilldie[i]로 선언한다. 만약 max와 whowilldie[i]의 값이 같을 경우는 표가 동일하게 나온 경우이므로 samecnt를 1로 지정한다.  401 ~ 402 : max값이 0보다 작은 경우는 이미 죽은 사람을 투표한 경우에 해당해서 -1을 반환 시켰다  404 ~ 408 : samecnt값이 1인 경우는 동표가 나온 경우라서 마찬가지로 -1을 반환  409 : 두 개의 case를 통과한 경우 maxindex를 반환 |

|  |
| --- |
| void become\_day() |
| 411 void become\_day(){  412 // 3. 낮이 됨  413 // 사회자가 마피아가 누구를 죽였다라고 말해줌  414 // 남아있는 사람들끼리 정해진 시간동안 토론을 한다.  415 // 정해진 시간이 지나면 한명 씩 지목한다.  416 // 사회자는 투표 중에서 동률이나 무효표가 아니면  417 // 그 사람을 처형하고  418 // 누군가 투표에 의해 죽었는지 알려준다.  419 // (종료조건을 확인)  420 printf("낮이 되었습니다.\n");  421  422 // 공유 메모리 만들기.  423 int shmid = shmget(sh\_key, MEMORY\_SIZE, IPC\_CREAT|0644);  424 if (shmid == -1) {  425 perror("shmget");  426 exit(1);  427 }  428  429 char day\_message[300] = "낮이 되었습니다.\n 마피아는 밤에 ";  430 strcat(day\_message, p\_nics[killnum]);  431 strcat(day\_message, "를 죽였습니다.\n정해진 ");  432 char tempstr[10]={0,};  433 sprintf(tempstr, "%d", discussion\_time);  434 strcat(day\_message, tempstr);  435 strcat(day\_message, "초 시간동안 대화를 시작해주세요. \n");  436 for(int i=0;i<client\_num;i++){  437 if(p\_jobs[i] == -1){  438 continue;  439 // 무시.. 죽었으니까.  440 }  441 send\_message(i+3, day\_message);  442 }  443  444 sleep(discussion\_time);  445 // 정해진 시간 기다리기.  446 printf("토론 시간이 종료되었습니다.\n 각 유저로 부터 I'm done메세지를 기다립니다.\n");  447 for(int i=0;i<client\_num;i++){  448 if(p\_jobs[i] == -1){  449 continue;  450 }  451 kill((pid\_t)p\_pids[i], SIGALRM);  452 }  453 // 토론이 끝났다는 시그널 날리기  454 // SIGALRM 메세지를 보낸다.  455  456 sleep(1);  457 // 1초 sleep  458  459  460 for(int i=0;i<client\_num;i++){  461 if(p\_jobs[i] == -1){  462 continue;  463 } // 죽은 아이디는 기다리지 않는다.  464 receive\_print\_message(1);  465 }  466 //I'm done 메세지 기다리기  467  468 shmctl(shmid, IPC\_RMID, NULL);  469 // 공유 메모리 삭제하기  470 printf("모든 done 메세지를 받았고, 공유 메모리를 삭제하였습니다. \n 투표를 진행합니다.\n");  471 // 투표 시작하기.  472 for(int i=0;i<client\_num;i++){  473 if(p\_jobs[i] == -1){  474 continue;  475 } // 죽은 아이디는 기다리지 않는다.  476 send\_message(i+3, "이제 투표를 시작합니다.! \n특정 유저의 번호를 지정해주세요!\n죽은사람은 투표시 무효표가 됩니다~!\n");  477 }  478  479 int whowilldie[client\_num];  480 memset(whowilldie, 0, sizeof(whowilldie));  481  482 printf("사용자들로 부터 표를 기다립니다.");  483 for(int i=0;i<client\_num;i++){  484 if(p\_jobs[i] == -1){  485 continue;  486 } // 죽은 아이디는 기다리지 않는다.  487 receive\_print\_message(1);  488 whowilldie[atoi(mesg.mtext)]++;  489 }  490  491 printf("투표가 성공적으로 치뤄졌습니다.\n");  492 killnum = vote\_result(whowilldie);  493 printf("투표 결과가 나옵니다.\n");  494 if(killnum != -1){  495 // 아래에서 죽이고 종료조건을 체크함.  496 print\_kill\_one\_check\_condition();  497 write\_user\_data();  498 // 선거 결과를 보내주기 전, 밤이 되기 전에 최신 데이터를 갱신.  499  500 char day\_killed\_message[300] = "투표가 종료되었습니다.\n 선거 결과\n ";  501 strcat(day\_killed\_message, p\_nics[killnum]);  502 strcat(day\_killed\_message, "를 죽였습니다.\n");  503  504 // 좀 이상하지만, 죽었음 알리고,  505 for(int i=0;i<client\_num;i++){  506 if(p\_jobs[i] == -1){  507 continue;  508 }  509 send\_message(i+3, day\_killed\_message);  510 if(p\_jobs[i] == Mafia){  511 memset(mesg.mtext, 0, sizeof(mesg.mtext));  512 sprintf(mesg.mtext, "%d", nopb\_jobs[Mafia]);  513 send\_message(i+3, mesg.mtext);  514 }  515 // 보내는 메세지 받는 사람이 마피아면 현재 마피아 수를 추가적으로 보내준다.  516 }  517  518 }  519 else{  520 write\_user\_data();  521 // 선거 결과를 보내주기 전, 밤이 되기 전에 최신 데이터를 갱신.  522  523 for(int i=0;i<client\_num;i++){  524 if(p\_jobs[i] == -1){  525 continue;  526 }  527 send\_message(i+3,"투표가 종료되었습니다. \n선거 결과 동률 또는 무효표로 인해 아무도 죽지 않았습니다.\n");  528 // 동률 또는 무효표에 의해서 아무도 죽지 않았음을 명시.  529 }  530 }  531  532 } |
| void become\_day() : 게임 상에서 낮을 구현한 함수  423 ~ 426 : 클라이언트 간의 채팅을 공유 메모리로 구현하기 위해 공유 메모리를 생성했다.  429 ~ 444 : 클라이언트에게 보낼 메시지인 day\_message라는 배열에 strcat를 이용해 낮이 되었음을 알리고 일정한 시간동안 대화를 시작하라는 내용을 저장하고 클라이언트에게 메시지를 보내고 토론시간 동안 sleep()을 한다.  446 ~ 453: 토론 시간이 종료되면 토론이 끝났다는 시그널을 pid를 통해 각각 클라이언트에게 전달한다.  460 ~ 465 : 각각의 클라이언트에게 타입 1의 메시지를 기다린다.  468 : 공유 메모리 삭제  472 ~ 477 : 죽은 클라이언트를 제외하고 투표를 시작한다고 메시지를 보낸다.  479 ~ 480 : 클라이언트의 득표를 whowilldie[]에 저장하고 낮이 올 때마다 값을 비워줘야 하므로 memset으로 초기화한다.  483 ~ 489 : 타입 1의 메시지 텍스트(플레이어 인덱스 번호)를 받고 whowilldie[atoi(mesg.mtext)]에 저장한다.  491 ~ 493 : vote\_result를 호출해 killnum에 저장한다  494 ~ 516: vote\_result의 반환 값이 -1이 아니면 투표로 죽을 사람이 정해진 것이다.  Print\_kill\_one\_check\_condition 함수로 제일 많이 득표한 사용자를 죽이고 write\_user\_date 함수로 최신 정보를 갱신한다. 그리고 클라이언트에게 메시지를 보내고 보내는 사람이 마피아면 현재 마피아의 수도 추가로 보낸다.  519~ 528 : killnum이 -1일 때는 투표로 누구도 죽지 않았음을 클라이언트에 보낸다. |

|  |
| --- |
| Main 함수 |
| 534 int main(void){  535 sh\_key= ftok("shmfile", 1);  536 // 공유 메모리를 위한 키 만들기.  537  538 // 1. 초기설정.  539 // 1-1. 접속 할 사람 수를 정하고, 사람 (process)의 접속을 기다린다.`  540 // 1-2. 사회자가 마피아를 랜덤으로 2명 선정  541 //1-3. 접속 정보를 저장하고 신호를 보내준다.  542 // (사회자 알려줘야 되요. 참여자 누구고 프로세스넘버 뭐있고, 닉네임뭐고.)  543  544 print\_start\_message();  545 scanf("%d", &client\_num);  546 printf("토론 시간을 입력 해주세요. : ");  547 scanf("%d", &discussion\_time);  548  549 printf("참여자의 접속을 기다립니다.\n");  550  551 send\_receive\_first\_message();  552 // 위 함수에서 1-1 뒷부분과과 1-2 를 실행.  553 // 1-1.뒷 부분. 사람 (process)의 접속을 기다린다.`  554 // 1-2. 사회자가 마피아를 랜덤으로 2명 선정  555  556 printf("test success\n");  557 printf("-------user\_data\_file------\n");  558 printf("--num nic pid -----\n");  559 for(int i=0;i<client\_num;i++){  560 if(p\_jobs[i] == -1){  561 continue;  562 }  563 printf("%d : %s %d\n", i, p\_nics[i], p\_pids[i]);  564 }  565 save\_user\_data\_send\_messages();  566 //1-3. 접속 정보를 저장하고 신호를 보내준다.  567 // (사회자 알려줘야 되요. 참여자 누구고 프로세스넘버 뭐있고, 닉네임뭐고.)  568  569 while(1){  570 // 2. 밤이 되었습니다.  571 // 마피아는 고개를 들어서 서로를 확인.  572 // 마피아 2명 이상이면 토의 시간 거치고  573 // 죽일 사람을 선택  574 // 그 자를 죽임.  575 // 종료 조건 확인.  576 become\_night();  577 // 3. 낮이 됨  578 // 사회자가 마피아가 누구를 죽였다라고 말해줌  579 // 남아있는 사람들끼리 정해진 시간동안 토론을 한다.  580 // 정해진 시간이 지나면 한명 씩 지목한다.  581 // 사회자는 투표 중에서 동률이나 무효표가 아니면  582 // 그 사람을 처형하고  583 // 누군가 투표에 의해 죽었는지 알려준다.  584 // (종료조건을 확인)  585 become\_day();  586 }  587 // 4. 2-3을 반복.  588 |
|  |

**-mg\_client.c**

|  |
| --- |
| 헤더 파일 추가 및 함수 선언 |
| 1 #include "mg\_header.h"  2  3 int mymtype;  4 char my\_nickname[30]={0,};  5 int myjobnum;  6 int mypid;  7 int killnum = -1;  8 // 내 정보(메세지 타입, 닉네임, 직업, 내 pid)  9 //내가 죽일 사람 번호.  10  11 int msgid;  12 struct mymsgbuf mesg;  13 char buf[BUFSIZ];  14 int n;  15 key\_t key;  16  17 // 메세지 관련 변수들.  18  19 char s\_message[500] = {0,};  20 struct sockaddr\_in sin, cli;  21 int sd, ns, clientlen = sizeof(cli);  22  23 struct sigaction act;  24 // sigaction 구조체 포인터 변수.  25 int discuss\_end\_flag = 0;  26 // 토론 종료를 판별하기 위한 플래그.  27 int mafia\_flag = 0;  28 // 마피아 플래그 0은 클라. 1은 서버.  29  30 FILE \*rfp, \*wfp;  31 // 파일 구조체 포인터  32  33 char \* nameofjobs[5]={"host","mafia","citizen",};  34 // 직업 명 문자열을 가리키는 포인터 배열.  35  36 char \* my\_data;  37 // 내가 보낼 초기 메세지.  38  39  40 int nopb\_jobs[5] = {0,};  41 // 직업별 사람수.  42  43 char pidstr[8] = {0,};  44 // pid를 문자열로 저장한 것.  45  46 void receive\_print\_message(int mtype);  47 void send\_message(int mtype, char \* message);  48  49 void make\_mynickname();  50 void send\_nickname\_receive\_jobnum();  51  52 void wait\_message\_read\_user\_data();  53  54 void receive\_message\_wait\_passing\_night();  55  56 void day\_discussion();  57 void receive\_message\_day\_discussion();  58 void receive\_message\_vote\_someone();  59  60 void end\_handler(int signo);  61  62 int input\_number\_condition\_check(); |
|  |

|  |
| --- |
| void receive\_print\_message(int mtype) void send\_message(int mtype, char \* message) |
| 66 void receive\_print\_message(int mtype){  67 int len = msgrcv(msgid, &mesg, 800, mtype, 0);  68 printf("Received Msg = %s \n", mesg.mtext);  69 }  70 void send\_message(int mtype, char \* message){  71 mesg.mtype = mtype;  72 strcpy(mesg.mtext, message);  73 if (msgsnd(msgid, (void \*)&mesg, 800, IPC\_NOWAIT) == -1) {  74 perror("msgsnd");  75 exit(1);  76 }  77 } |
| mg\_server.c에서 설명한 것과 동일 |

|  |
| --- |
| void make\_mynickname() |
| 79 void make\_mynickname(){  80 printf("안녕하세요 마피아 게임입니다.\n");  81 printf("당신은 게임의 참여자로 선정되었습니다.\n");  82 printf("제작 자 : 강대훈, 이동석.\n");  83 printf("당신의 닉네임을 정해주세요!\n");  84 printf("입력(특수문자 제외 20자이내) : ");  85  86 my\_data = malloc(sizeof(char) \* 30);  87 scanf("%s", my\_data);  88 //printf("네, 당신의 nickname은 %s입니다!\n", my\_data);  89  90 strcpy(my\_nickname, my\_data);  91 mypid = (int)getpid();  92 sprintf(pidstr, "%d", mypid);  93 //printf("그리고 당신의 pid은 %s입니다!\n", pidstr);  94  95 strcat(my\_data, "-");  96 strcat(my\_data, pidstr);  97 //printf("그리고 당신의 메세지는 %s입니다!\n", my\_data);  98  99 // 닉네임을 입력 받고, 자신의 pid와 합쳐서 메세지를 만드는 과정.  100 } |
| void make\_mynickname() : 닉네임을 하고 받고, 자신의 pid와 합쳐서 메시지를 만드는 과정  90 : getpid 함수를 사용해 process 번호를 mypid에 저장 |

|  |
| --- |
| void send\_nickname\_receive\_jobnum() |
| 102 void send\_nickname\_receive\_jobnum(){  103 key = ftok("mg", 1);  104 // key 값을 생성합니다.  105 if((msgid = msgget(key, 0)) < 0){  106 perror("msgget");  107 exit(1);  108 }  109 // key값과 IPC\_CREAT은 새로운 키면 식별자를 새로 생성합니다. 그렇지 않으면 새로 생성하지 않습니다.  110 // 0644 는 펄미션입니다. 110 100 100  111 // 이것은 .. 그렇습니다.  112 // 반환하는 것은 message identifier 메세지 식별자 입니다.  113  114 send\_message(1, my\_data);  115 receive\_print\_message(2);  116 // 만들어둔 닉네임과 pid를 보냅니다.  117  118 mymtype = atoi(mesg.mtext);  119 //printf("너의 mymtype : %d \n", mymtype);  120 // 클라이언트가 쓸 mtype을 저장.  121  122 receive\_print\_message(mymtype);  123 myjobnum = atoi(mesg.mtext);  124 //printf("너의 myjobnum : %d \n", myjobnum);  125 //참여자의 직업 번호를 저장.  126  127 if(myjobnum == Mafia){  128 receive\_print\_message(mymtype);  129 // 마피아일 경우에는 마피아 수를 추가적으로 받음.  130 nopb\_jobs[Mafia] = atoi(mesg.mtext);  131 printf("현재 마피아 수 : %d\n", nopb\_jobs[Mafia]);  132 // 여기 뒤에 마피아가 2명 이상일 때, 서버역할인지 클라이언트  133 //// 1번째 정보.  134  135 if(nopb\_jobs[Mafia] >=2){  136 receive\_print\_message(mymtype);  137 mafia\_flag = atoi(mesg.mtext);  138 // 2번째는 마피아 클라인가 서버인가?  139  140 receive\_print\_message(mymtype);  141 // 3번째는 마피아 정보.  142 }  147 }  148 } |
| void send\_nickname\_receive\_jobnum() : 닉네임을 서버에 보내고 직업 번호를 받는 함수  105 ~ 108 : 메시지 큐 생성  114 : my\_data를 서버에 보냄.  118 : 서버에서 받은 메시지 타입을 정수형으로 변환한 뒤 mymtype에 저장  122 ~ 123 : 서버로부터 직업 번호를 받는다.  127 ~ 142 : 만약 마피아일 경우에는 서버로부터 마피아의 수와 정보를 추가적으로 받는다. 맨 처음 서버에서 마피아 정보를 받는 클라이언트는 mafia\_flag가 0인 값을 받는다. |

|  |
| --- |
| void print\_your\_data\_read\_user\_data() |
| 150 void print\_your\_data\_read\_user\_data(){  151  152 printf("\n\n--------------------------------\n");  153 printf("당신이 현재 알고 있는 정보들.\n");  154 printf("--------------------------------\n");  155 printf("당신의 참여자 번호는 %d입니다.\n", mymtype -3);  156 printf("당신의 닉네임은 %s입니다.\n", my\_nickname);  157 printf("당신의 pid는 %d입니다.\n", mypid);  158 printf("당신의 직업은 %s입니다.\n", nameofjobs[myjobnum]);  159  160 if ((rfp = fopen("user\_data.txt", "r")) == NULL) {  161 perror("fopen: user\_data.txt");  162 exit(1);  163 }  164 while ((n = fread(buf, sizeof(char), 1024, rfp)) > 0) {  166 printf("%s", buf);  167 }  168 fclose(rfp);  169 printf("\n");  170 // 유저 정보를 읽어 출력합니다.  171 } |
| void print\_your\_data\_read\_user\_data() : 현재 클라이언트의 정보를 화면에 출력하고 추가로 user\_data.txt에 저장된 파일을 읽기 모드로 열어서 유저 정보를 출력한다. |

|  |
| --- |
| void wait\_message\_read\_user\_data(),int input\_number\_condition\_check\_2(char input [30]) |
| 173 void wait\_message\_read\_user\_data(){  174 receive\_print\_message(mymtype);  175 // 메세지를 받습니다.  176 print\_your\_data\_read\_user\_data();  177  178 }  179  180 int input\_number\_condition\_check\_2(char input [30]){  181 //printf("%d\n", strlen(input));  182 if(strlen(input) !=2){  183 return -1;  184 }  185 if ( '0' <= input[0] && input[0] <= '9')  186 {  187 int num = input[0] - '0';  188 //printf("성공 \n");  189 return num;  190 }  191 else  192 return -1;  193 }  194 |
| void wait\_message\_read\_user\_data() : 메시지를 받고 정보를 화면에 출력해주는 함수  int input\_number\_condition\_check\_2(char input [30]) : 입력 받은 문자의 길이가 개행을 포함해해 2가 아니면 -1을 반환하고, 입력 받은 것이 0과 9 사이의 숫자면 num이라는 변수에 입력 받은 수를 대입하고 num을 반환한다. |

|  |
| --- |
| void socket\_client() |
| 195 void socket\_client(){  196 if ((sd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) {  197 perror("socket");  198 exit(1);  199 }  200  201 memset((char \*)&sin, '\0', sizeof(sin));  202 sin.sin\_family = AF\_INET;  203 sin.sin\_port = htons(PORTNUM);  204 sin.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  205  206 if (connect(sd, (struct sockaddr \*)&sin, sizeof(sin))) { // 서버에 연결 요청함.  207 perror("connect");  208 exit(1);  209 }  210 while(1){  211 printf("메세지를 기다립니다...\n");  212 if (recv(sd, s\_message, sizeof(s\_message), 0) == -1) {  213 perror("recv");  214 exit(1);  215 }  216 printf("From Server : %s\n", s\_message);  217 if(input\_number\_condition\_check\_2(s\_message) >= 0){  218 break;  219 }  220 printf("할말을 입력하세요.(0이상 9이하의 한자리 숫자를 받으면 종료됩니다.): ");  221 fgets(s\_message, sizeof(s\_message), stdin);  222  223 if (send(sd, s\_message, strlen(s\_message) + 1, 0) == -1) {  224 perror("send");  225 exit(1);  226 }  227 if(input\_number\_condition\_check\_2(s\_message) >= 0){  228 break;  229 }  230 }  231 printf("합의된 죽일 사람 번호 : %d\n", atoi(s\_message));  232 //test  233 close(sd);  234  235 } |
| void socket\_client() : 마피아가 통신하는 방식을 소켓으로 구현함. (클라이언트)  196 ~ 199 : 소켓 종류는 AF\_INET, type은 SOCK\_STREAM, 프로토콜은 0으로 소켓을 생성  201 ~ 204 : 소켓 구조체에 소켓 종류, port번호, ip 주소를 대입  206 ~ 208 : 서버 마피아에게 연결 요청을 한다.  210 ~ 230 : 클라이언트와 서버 사이 연결이 되면 recv, send 함수로 메시지를 주고 받음. 클라이언트는 먼저 서버 메시지를 받고 출력한 다음 fgets로 말을 입력하고 입력한 문자열을 send 함수로 서버에게 보낸다. Recv, send 함수 바로 밑에 input\_number\_condition\_check2 함수를 호출하여 받거나 보낸 문자열이 0이상 9이하 수면 반복문을 빠져나간다.  231 ~ 233 : 죽일 사람을 입력하면 출력하고 소켓을 닫는다. |

|  |
| --- |
| void socket\_server() |
| 236 void socket\_server(){  237  238 if ((sd = socket(AF\_INET, SOCK\_STREAM, 0)) == -1) { // 인터넷 소켓 생성.  239 perror("socket");  240 exit(1);  241 }  242  243 memset((char \*)&sin, '\0', sizeof(sin)); //  244 sin.sin\_family = AF\_INET;  245 sin.sin\_port = htons(PORTNUM);  246 sin.sin\_addr.s\_addr = inet\_addr("127.0.0.1");  247  248 if (bind(sd, (struct sockaddr \*)&sin, sizeof(sin))) {  249 perror("bind");  250 exit(1);  251 }  252 if (listen(sd, 5)) { // 클라이언트의 연결을 기다림 (최대 5개)  253 perror("listen");  254 exit(1);  255 }  256  257  258 if ((ns = accept(sd, (struct sockaddr \*)&cli, &clientlen))==-1) { // 클라이언트의 접속을 수용함.  259 perror("accept");  260 exit(1);  261 }  262 printf("connected\n");  263 //sprintf(buf, "Your IP address is %s", inet\_ntoa(cli.sin\_addr));  264  265 while(1){  266 printf("할말을 입력하세요.(0이상 9이하의 한자리 숫자를 받으면 종료됩니다.): ");  267 fgets(s\_message, sizeof(s\_message), stdin);  268 //scanf("%s",s\_message);  269  270 if (send(ns, s\_message, strlen(s\_message) + 1, 0) == -1) {  271 perror("send");  272 exit(1);  273 }  274 if(input\_number\_condition\_check\_2(s\_message) >= 0){  275 break;  276 }  277 printf("메세지를 기다립니다...\n");  278 if (recv(ns, s\_message, sizeof(s\_message), 0) == -1) {  279 perror("recv");  280 exit(1);  281 }  282 printf("From Client : %s\n", s\_message);  283 if(input\_number\_condition\_check\_2(s\_message) >= 0){  284 break;  285 }  286 }  287 printf("합의된 죽일 사람 번호 : %d\n", atoi(s\_message));  288  289  290 close(ns);  291 close(sd);  292  293  294 } |
| void socket\_server() : 마피아가 통신하는 방식을 소켓으로 구현함. (서버)  243 ~ 246 : 클라이언트와 마찬가지로 소켓 구조체에 값을 채움.  248 ~ 250 : bind 함수로 소켓에 포트를 지정  252 ~ 255 : listen 함수로 클라이언트의 연결을 기다림  258 ~ 261 : accept 함수로 클라이언트의 접속을 수용함.  265 ~ 291 : 클라이언트와 설명 동일. 단 서버는 먼저 클라이언트에게 메시지를 보내고 메시지를 받는다. |

|  |
| --- |
| 297 void receive\_message\_wait\_passing\_night() |
| 297 void receive\_message\_wait\_passing\_night(){  298 receive\_print\_message(mymtype);  299 // clear?  300 print\_your\_data\_read\_user\_data();  301 if(myjobnum == Mafia){  302 if(nopb\_jobs[Mafia] ==1){  303 printf("마피아를 제외하고 죽일 상대의 num을 입력하세요.\n");  304 killnum = input\_number\_condition\_check();  305 memset(mesg.mtext, 0, sizeof(mesg.mtext));  306 sprintf(mesg.mtext, "%d", killnum);  307 send\_message(1, mesg.mtext);  308 }  309 else if(nopb\_jobs[Mafia] ==2){  310 if(mafia\_flag == 0){  311 sleep(1);  312 socket\_client();  313 }  314 else if( mafia\_flag ==1){  315 socket\_server();  316 }  317 send\_message(1,s\_message);  318 }  319 }  320 //마피아만 죽일 상대를 정해 사회자에게 알려준다.  321 // 여기서 2명 이상일 때 소켓통신 해야할듯.  322 // 다른 참여자는 낮이 되길 기다린다.건  323  324 if(myjobnum ==Mafia){  325 receive\_print\_message(mymtype);  326 nopb\_jobs[Mafia] =atoi(mesg.mtext);  327 // 마피아는 추가적으로 메세지를 한번 더 받는다.  328 }  329 } |
| void receive\_message\_wait\_passing\_night() : 자신의 직업이 마피아일 경우 소켓 통신으로 죽일 상대를 정하는 함수.  298 :  301 ~ 308 : 마피아가 1명일 땐 input\_number\_condition\_check()에서 나온 값을 killnum에 저장하고 서버에 killnum을 보낸다.  309 ~ 329 : 마피아가 2명일 때는 mafia\_flag가 0인 마피아는 socket\_client를 실행하고 1인 마피아는 socket\_server 함수를 실행하고 메시지를 보낸다.  324 ~ 329 : 마피아는 추가로 서버에서 주는 마피아의 수를 받는데 이를 처리하는 코드 |

|  |
| --- |
| int input\_number\_condition\_check() |
| 331 int input\_number\_condition\_check(){  332 char input [30];  333 while (1) {  334 printf("입력 : ");  335 scanf("%s", input);  336  337 if(strlen(input) !=1){  338 printf("조건에 맞지 않습니다. 다시 입력하세요. \n");  339 continue;  340 }  341 if (strcmp(input, "0")==0)  342 {  343 //printf("성공 \n");  344 return 0;  345 }  346 else if (atoi(input) > 0 && atoi(input) <= 9)  347 {  348 int num = atoi(input);  349 //printf("성공 \n");  350 return num;  351 }  352 else  353 {  354 printf("조건에 맞지 않습니다. 다시 입력하세요. \n");  355 continue;  356 }  357 }  358 } |
| int input\_number\_condition\_check() : 번호 입력 조건 함수 0~9 이외의 값이 나오면 다시 입력하라고 처리. Atoi 함수는 정수 반환 실패시 0을 반환하므로 341번째 줄에 있는 조건처럼 0을 따로 처리해야한다. |

|  |
| --- |
| void day\_discussion() |
| 360 void day\_discussion(){  361 key\_t sh\_key = ftok("shmfile", 1);  362 int shmid = shmget(key, MEMORY\_SIZE, 0);  363 char \* shamddr = (char\*)shmat(shmid,(char \*)NULL, 0); // shared memory 연결  364 while(1)  365 {  366 if(discuss\_end\_flag ==1){  367 printf("토론이 종료되었습니다.\n");  368 printf("잠시 기다려주세요.\n");  369 discuss\_end\_flag = 0;  370 break;  371 }  372 int a;  373 printf("어떤 행동 할래?\n");  374 printf("1 : 메세지 날리기.\n");  375 printf("2 : 메세지 보기.\n");  376  377 scanf("%d", &a);  378 getchar();  379  380 if (a==1)  381 {  382  383 char str1[1000] = {0,};  384 printf("메세지를 입력하세요.\n입력 : ");  385 //scanf("%s", str1);  386 fgets(str1,sizeof(str1), stdin);  387 ////fgets(buf, sizeof(buf), stdin);  388  389 strcat(shamddr, my\_nickname);  390 shamddr[strlen(shamddr)] = ':';  391 strcat(shamddr, str1);  392 shamddr[strlen(shamddr)] = '\n';  393 // shmdt((char \*)shmaddr); // shared memory 연결 해제  394 }  395 else if(a==2)  396 {  397 printf("===================================\n");  398 printf("%s", shamddr); // shared memory에 있는 내용을 출력  399 printf("===================================\n");  400  401 }  402 }  403 // 공유 메모리를 이요함.  404 // 입력이 1이면 메세지를 입력.  405 // 입력이 2면 지금 까지 있는 메세지를 보여줌.  406  407  408 shmdt(shamddr);  409  410 send\_message(1, "I'm done\n");  411 // 정해진 시간이 끝났을 떄 나 끝났다는 메세지를 보냄.  412 } |
| void day\_discussion() : 낮이 되었을 때 플레이어 끼리 공유 메모리를 통해 소통하게 만드는 함수  361 ~ 363 : 서버에 만들어져 있던 공유 메모리 연결  365 ~ 369 : 토론이 종료되면 discuss\_end\_flag가 0이 되면서 반복문을 빠져나감  373 ~ 379 : 1을 입력하면 메시지를 입력, 2를 입력하면 메시지를 볼 수 있다.  380 ~ 392 : 메시지를 입력받고 공유된 메모리에 입력받은 문자열을 넣는다.  395 ~ 402 : 공유 메모리에 들어있는 내용을 출력  408, 410 : 토론 시간이 끝나면 공유 메모리를 해제하고 서버에게 끝났다는 메시지를 보낸다. |

|  |
| --- |
| void receive\_message\_day\_discussion() |
| 414 void receive\_message\_day\_discussion(){  415 receive\_print\_message(mymtype);  416 // clear?  417 print\_your\_data\_read\_user\_data();  418 day\_discussion();  419 } |
| void receive\_message\_day\_discussion() : 위에서 설명한 3개의 함수를 하나의 모듈로 만듬 |

**‘**

|  |
| --- |
| void end\_handler(int signo) |
| 421 void end\_handler(int signo){  422 if(signo == SIGUSR2){  423 printf("시민은 승리하였습니다!\n");  424 }  425  426 if(signo == SIGUSR1){  427 printf("마피아가 승리하였습니다!\n");  428 }  429  430 if(signo == SIGQUIT){  431 printf("당신은 죽임을 당했습니다. \n");  432 printf("---GameOver..---\n");  433  434 }  435  436  437 if(signo == SIGALRM){  438 discuss\_end\_flag = 1;  439 }  440  441 if(signo ==SIGUSR1 || signo == SIGUSR2){  442 if ((rfp = fopen("user\_data.txt", "r")) == NULL) {  443 perror("fopen: user\_data.txt");  444 exit(1);  445 }  446  447 while ((n = fread(buf, sizeof(char), 1024, rfp)) > 0) {  448 //fwrite(buf, sizeof(char), n, 1);  449 printf("%s", buf);  450 }  451 fclose(rfp);  452 printf("\n");  453 }  454  455  456 if(signo == SIGUSR1 && signo ==SIGUSR2 && signo ==SIGQUIT){  457 close(ns);  458 close(sd);  459 // 종료 할때, 소켓을 닫아주자.  460 }  461 // 결과 공개.  462 if(signo != SIGALRM){  463 exit(1);  464 }  465 } |
| void end\_handler(int signo) : 각각의 시그널을 처리하는 핸들러  422 ~ 424 : 시민의 승리 시그널을 처리하는 핸들러  426 ~ 428 : 마피아의 승리 시그널을 처리하는 핸들러  430 ~ 433 : 플레이어가 죽임을 당했을 때의 시그널을 처리하는 핸들러  437 ~ 439 : 토론 시간이 끝나면 discuss\_end\_flag를 1로 처리하는 핸들러  441 ~ 453 : 시민 또는 마피아 승리 시 사람들의 직업, 닉네임 등 프로필이 저장되어 있는 파일을 읽기 모드로 열고 출력 해주는 핸들러  456 ~ 458 : 게임이 끝날 때 소켓을 닫아줌  462 ~ 464 : SIGALRM이 아닌 시그널이 오면 프로그램 종료 |

|  |
| --- |
| void receive\_message\_vote\_someone() |
| 467 void receive\_message\_vote\_someone(){  468 receive\_print\_message(mymtype);  469 // clear?  470 print\_your\_data\_read\_user\_data();  471  472 printf("특정 유저의 번호를 지목해주세요. \n");  473 memset(mesg.mtext, 0, sizeof(mesg.mtext));  474 int selectnum = input\_number\_condition\_check();  475 sprintf(mesg.mtext, "%d", selectnum);  476 send\_message(1, mesg.mtext);  477 // 특정 유저의 번호를 입력 받아 그대로 사회자에게 전달.  478 } |
| void receive\_message\_vote\_someone() : 서버에서 메시지가 오기를 기다리다가 특정 유저의 번호를 지목해 서버에게 전달하는 함수 |

|  |
| --- |
| Main 함수 |
| 480 int main(void) {  481  482  483 act.sa\_flags = 0; // 아무것도 설정안한 것.  484 act.sa\_handler = end\_handler; //핸들러로 위에 것을 지정한것.  485 if (sigaction(SIGUSR2, &act, (struct sigaction \*)NULL) < 0) {  486 perror("sigaction");  487 exit(1);  488 }  489 if (sigaction(SIGUSR1, &act, (struct sigaction \*)NULL) < 0) {  490 perror("sigaction");  491 exit(1);  492 }  493 if (sigaction(SIGQUIT, &act, (struct sigaction \*)NULL) < 0) {  494 perror("sigaction");  495 exit(1);  496 }  497 if (sigaction(SIGALRM, &act, (struct sigaction \*)NULL) < 0) {  498 perror("sigaction");  499 exit(1);  500 }  501 //signal 설정  502 //client 프로그램 시작.  503  504 // 1. 초기설정.  505 // 1-1. 자신의 닉네임 정하기 닉네임 보내고 직업정보 받기.  506 // 1-2. 메시지 기다리고 유저 정보 확인하기  507  508 make\_mynickname();  509 send\_nickname\_receive\_jobnum();  510 // 1-1. 자신의 닉네임 정하기 닉네임 보내고 직업정보 받기.  511  512  513 wait\_message\_read\_user\_data();  514 // 1-2. 메시지 기다리고 유저 정보 확인하기  515 while (1){  516 // (밤)  517 // 2. 메세지 받고 밤이 지나길 기다리기  518 receive\_message\_wait\_passing\_night();  519  520 // (낮)  521 // 3. 메세지를 받고 낮에 토론하기  522 receive\_message\_day\_discussion();  523  524  525 //4.투표하고 투표결과 받기.  526 receive\_message\_vote\_someone();  527 receive\_print\_message(mymtype);  528 if(myjobnum ==Mafia){  529 receive\_print\_message(mymtype);  530 nopb\_jobs[Mafia] =atoi(mesg.mtext);  531 // 마피아는 추가적으로 메세지를 한번 더 받는다.  532 }  533 // 투표 결과를 받음.  534  535 }  536 //5. 2-4를 반복하기.  537  538  539 return 0;  540 } |
|  |

**-**

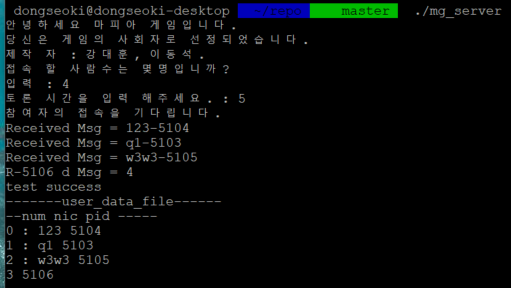
**mg\_header.h**

|  |
| --- |
| Mg\_header.h |
| 1 #include <signal.h>  2 #include <unistd.h>  3  4 #include <string.h>  5 #include <stdlib.h>  6 #include <stdio.h>  7  8 #include <sys/ipc.h>  9 #include <sys/mman.h>  10 #include <sys/msg.h>  11 #include <sys/shm.h>  12 #include <sys/stat.h>  13 #include <sys/types.h>  14 #include <sys/socket.h>  15 #include <arpa/inet.h>  16 #include <time.h>  17  18 #define MAX\_PEOPLE 10  19 #define MEMORY\_SIZE 10000  20 #define PORTNUM 9004  21  22 struct mymsgbuf {  23 long mtype;  24 char mtext[800];  25 };  26  27  28 enum Jobs{  29 Host =0,  30 Mafia,  31 Citizen  32 };  ~ |
| 1 ~ 16 : 헤더 선언  22 ~ 25 : 메시지 구조체 선언  28 ~ 32 : 열거형을 통해 host =0, mafia = 1, citizen = 2로 선언 |

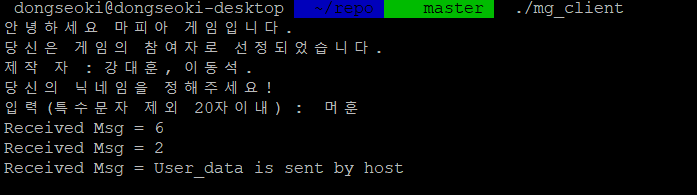
# 테스트 결과

- 게임 시작.

1) 사회자는 서버

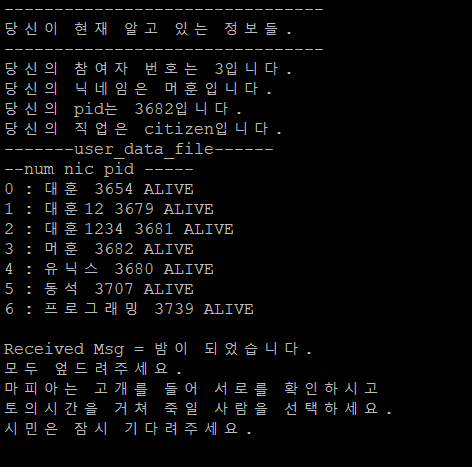


2) 플레이어는 클라이언트

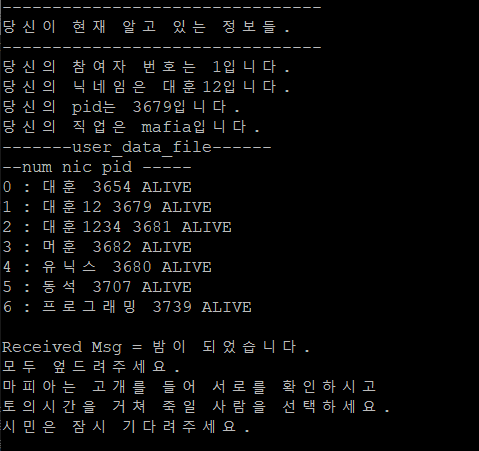


- 모든 클라이언트가 접속을 하면 랜덤으로 직업 배정

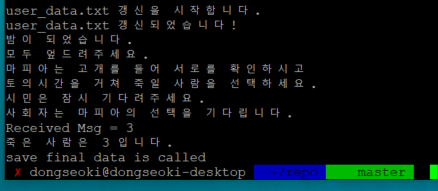
1) 시민

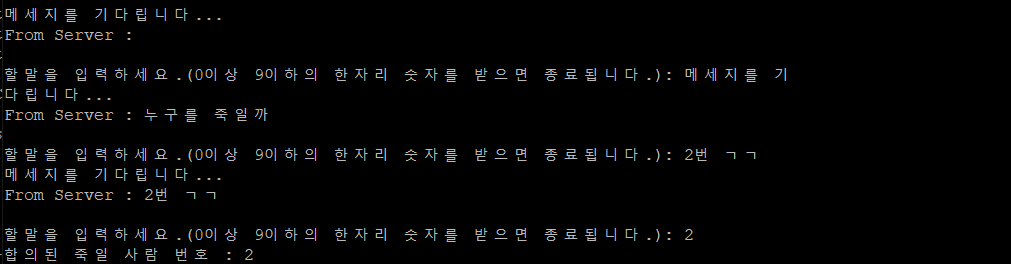


2) 마피아



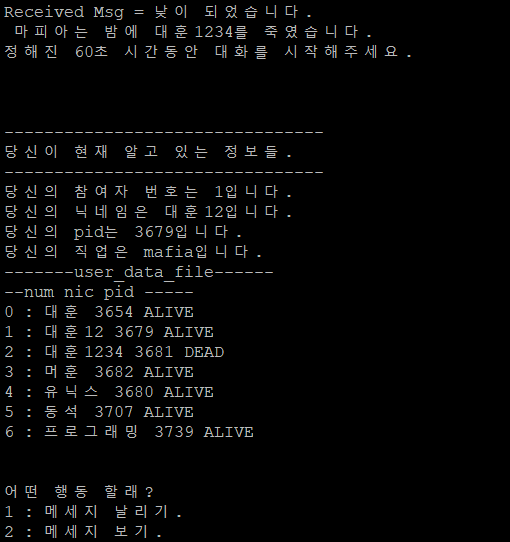
- 밤이 된 다음 마피아들의 소켓 통신



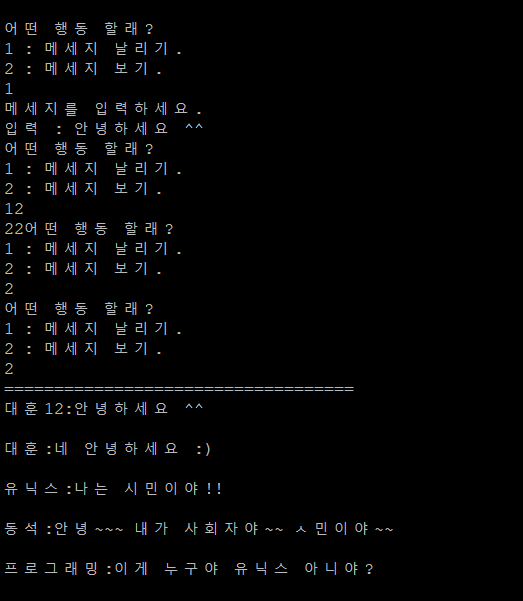


- 낮이 된 다음 공유 메모리를 이용한 사용자들의 통신

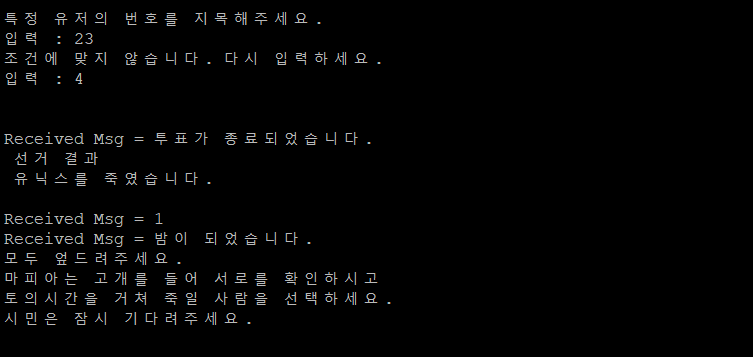
1) 낮



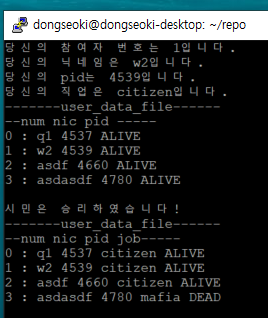
2) 메시지 보내기



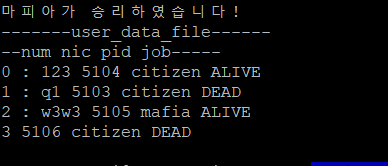
- 토론이 끝난 다음 투표 진행



- 시민 승리



- 마피아 승리



# 추후 발전 방향

- 본 프로그램에서는 직업이 마피아와 시민밖에 없지만 의사나 경찰 등 다양한 직업의 역할을 구현하여 게임의 재미를 더욱 유발할 수 있다.

- 플레이 스토어에서 올라온 마피아 게임처럼 UI, 서버, database 기술을 이용하여 더욱 완성도가 높은 게임을 만들 수 있다.

- 두 자리 숫자를 입력 받을 경우 예외 처리를 해서 최대 9명 까지 게임 이용이 가능하다. 10명 이상의 사용자도 게임이 가능하게 발전 시킬 수 있다.

- 서버와 클라이언트 간의 소켓 통신을 이용해 원격으로도 게임을 가능하게 할 수 있다.