컴퓨터네트워크

김태운

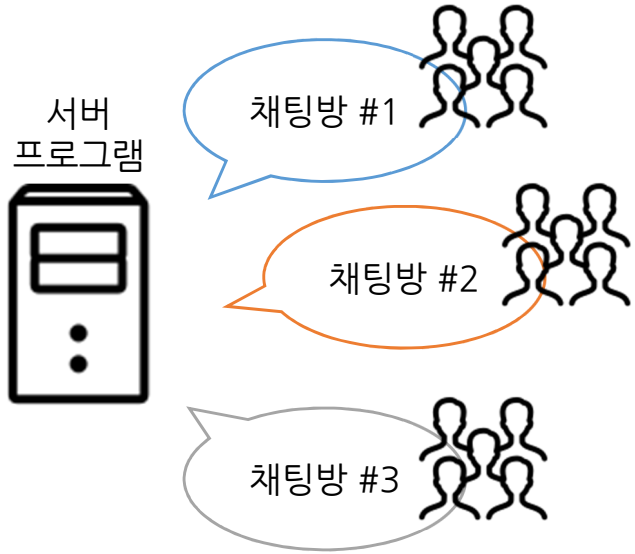
소켓프로그래밍 4: 과제 설명

실습과제 소개

- 소켓 프로그램 4 (마지막)
  - 제출물 : 보고서, 소스코드, 실행 동영상 (데모 시나리오를 따라서 실행하고, 화면 녹화하여 제출)
    - 각각 미제출 시 감점 있음
  - 마지막 과제이며 200점 만점
  - [데모 시나리오] 페이지에 명시된 순서에 따라 모든 과정이 정상적으로 진행되어야 함

- 동영상 촬영 시 주의사항
  - 데모 시나리오에 따라서 실행하는 화면을 녹화
    - 총 9개 터미널(서버 프로그램 터미널 1개 + 4명 클라이언트 터미널 총 8개) 화면이 모두 녹화되어야 함
    - 학생의 얼굴/목소리 등은 녹화영상에 포함할 필요 없음
    - 단, 지정된 시나리오를 따라서 천천히 실행할 것
  - 녹화 화면의 해상도가 충분히 높아야 하며, 모든 터미널에 출력된 모든 문자를 읽을 수 있는 수준으로 선명해야 함

채팅 프로그램, *final*



- 아래의 요구사항을 만족하는 채팅 프로그램을 작성하시오
  - 서버에서 다수의 채팅 방을 개설할 수 있음
  - 클라이언트는 하나의 채팅 방에만 참여할 수 있음
  - 하나의 채팅 방에는 다수의 사용자가 참여할 수 있음
  - 사용자는 비 동기형 채팅이 가능해야 함 (즉, 연속 send 가 가능해야 함)

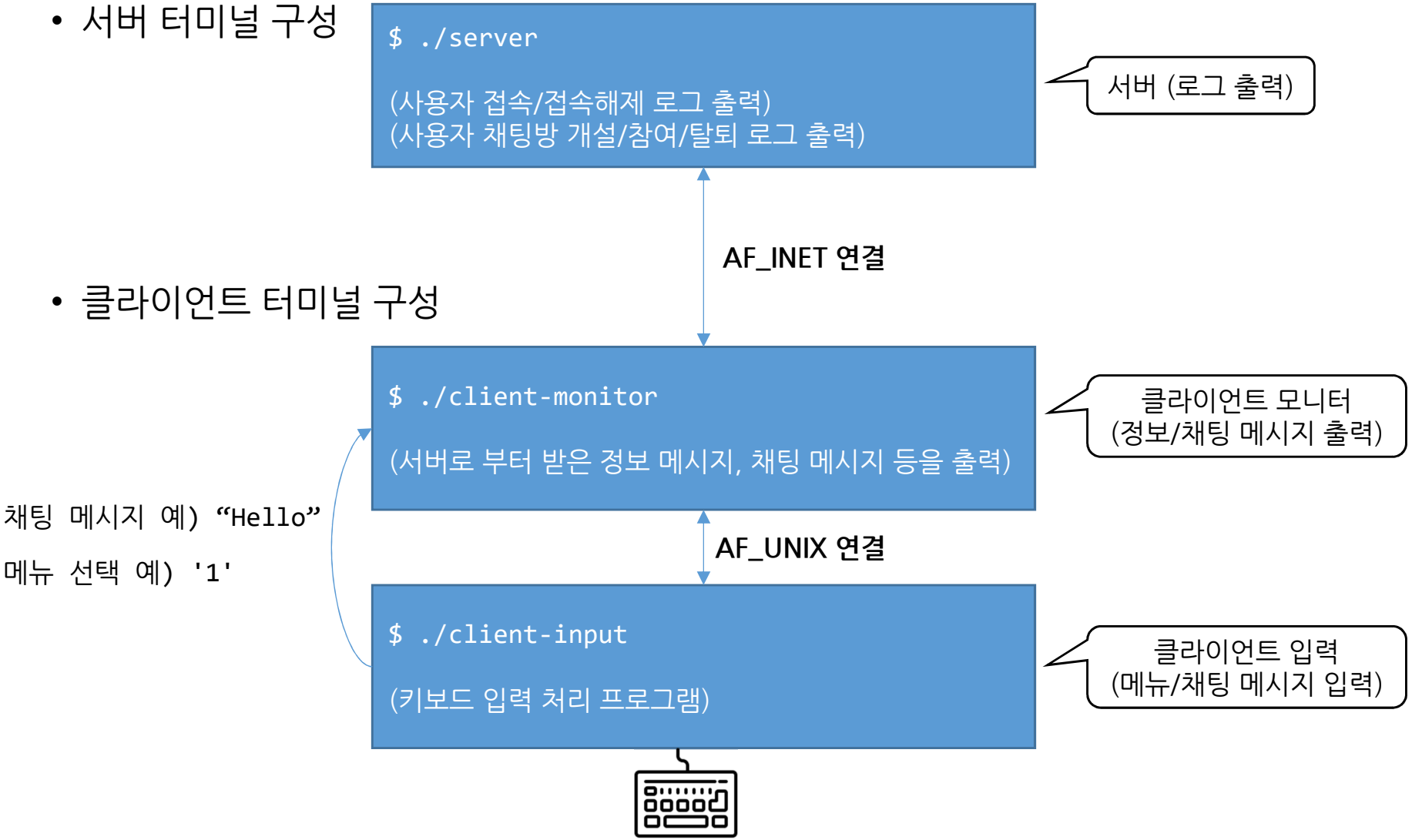
  - 서버 프로그램
    - 서버 프로그램에서는 어떠한 메시지도 입력하지 않음 (즉, 터미널 입력을 받지 않음)
    - 서버 프로그램은 아래의 정보를 터미널에 출력해야 함 (서버 터미널에 출력 O. 클라이언트에 전송 X)
      - 사용자 접속(connect 요청)시 및 접속 해제 시 로그 메시지 출력
      - 사용자의 채팅방 개설/참여/탈퇴 시 로그 메시지 출력
    - 클라이언트 프로그램은 두 개의 터미널로 구성 :
      - 모니터 터미널
        - 서버가 보내주는 정보 메시지, 채팅 메시지 등이 출력되는 터미널 화면
        - 출력 전용 터미널이며, 키보드 입력을 받지 않음
      - 입력 터미널
        - 키보드 입력을 위한 터미널
        - 입력 전용 터미널

어떤 이벤트가 일어났는지에 대한 로그를 남기기 위함 입니다. 클라이언트에게 전송하는 메시지가 아닙니다.

채팅 프로그램, *final*

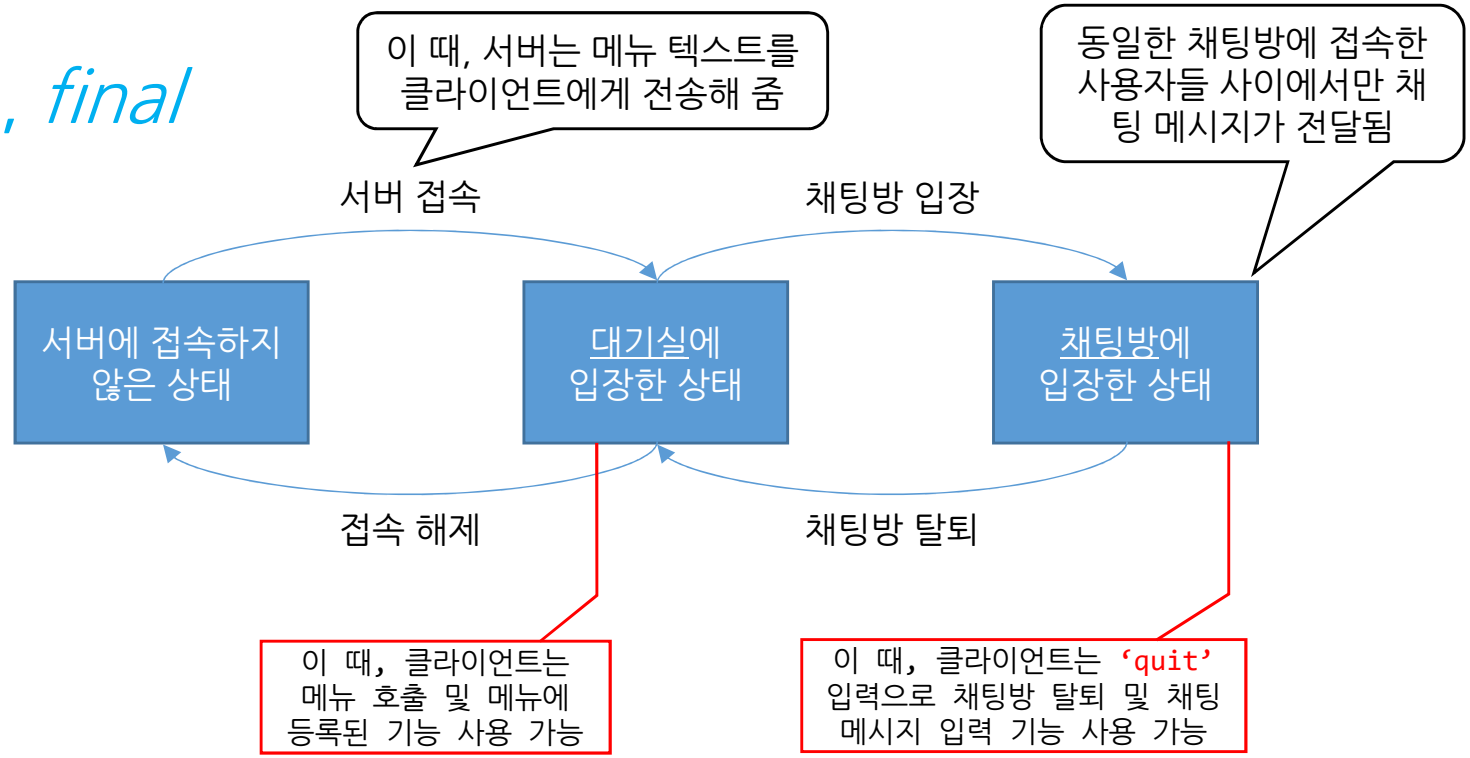
같은 채팅방에 있는 사용자는 서로 채팅을 할 수 있는데, 이 때 채팅 메시지는 P2P로 전송되는 것이 아니고, 서버를 경우 하여 다른 사용자에게 전달됨

- 서버 터미널 구성
- 클라이언트 터미널 구성



채팅 프로그램, *final*

- 클라이언트 상태 및 상태 별 가용 기능



- 클라이언트 행동

<클라이언트가 대기실에서 할 수 있는 행동>
- '0'번을 입력하여 서버로 전송하면, 서버는 메뉴 텍스트를 회신해줌
- '1'번을 입력하여 서버로 전송하면, 서버는 개설된 채팅방 목록(고유번호, 제목, 참여자 수) 텍스트를 회신해줌
- '2'번과 채팅방 고유번호를 입력하면(예: 2 1) 해당 채팅방에 입장
- '3'번을 입력하여 서버로 전송하면, 서버와의 접속을 해제함
- 이 외에는 잘못된 명령어를 입력한 것으로 처리함

<클라이언트가 채팅방에서 할 수 있는 행동>
- 'quit'을 입력하면 채팅방에서 탈퇴하여 대기실로 돌아 감
- 이 외에는 모두 채팅 메시지로 간주하고, 같은 채팅방에 있는 다른 모든 사용자에게 메시지를 전송함

채팅 프로그램, *final*

- 서버 기능
  - 동시 동작 서버로 동작하며, 다수의 채팅 방 및 다수의 클라이언트를 동시에 병렬적으로 서비스 할 수 있어야 함
  - 모든 클라이언트는 비동기 채팅이 가능해야 함(즉, 클라이언트는 언제든 원하는 시점에 메시지를 보낼 수 있어야 함)
  - 클라이언트의 connect 요청에 accept로 응답한 직후, 클라이언트에게 메뉴 텍스트를 전송함
- 메뉴 텍스트
  - 클라이언트가 서버에 접속 시, 또는 대기실에서 '0'번을 서버로 전송 시 아래와 같은 메뉴 텍스트를 클라이언트에게 전송함

```
<MENU>
1.채팅방 목록 보기
2.채팅방 참여하기(사용법: 2 <채팅방 번호>)
3.프로그램 종료
(0을 입력하면 메뉴가 다시 표시됩니다)
```

- 참고: 한글 지원이 안될 경우, 영어로 작성해도 됨(메뉴, 로그 메시지, 채팅 메시지 등)

채팅 프로그램, *final*

클라이언트와의 통신을 위한 소켓 기술자의 번호(int)를 해당 클라이언트의 고유 ID인 것 처럼 사용합니다. 소켓 번호는 상황에 따라 다른 번호가 할당 되며, 화면에 보이는 것과는 다른 숫자가 할당 될 수 있습니다.

- 서버 프로그램 터미널은 아래와 같은 정보 메시지(= 로그)를 출력해야 함
  - 정보 메시지는 관리자를 위해 출력하는 메시지(또는 로그기록)이며, 사용자에게 전달되지 않음

```
[MAIN] 새로운 사용자가 접속했습니다 : 4
[MAIN] 새로운 사용자가 접속했습니다 : 5
[MAIN] 사용자 4 메시지 : 0
[MAIN] 사용자 4 메시지 : 1
[MAIN] 사용자 5 메시지 : 1
[MAIN] 사용자 5 메시지 : 2 0
[MAIN] 사용자 5가 채팅방 0 에 참여합니다
[Ch.0] 새로운 참가자 : 5
[MAIN] 사용자 4 메시지 : 2 0
[MAIN] 사용자 4가 채팅방 0 에 참여합니다
[Ch.0] 새로운 참가자 : 4
[Ch.0] 사용자 4 의 메시지 : hello world~
[Ch.0] 사용자 4의 메시지를 전달합니다.
[Ch.0] 사용자 5 의 메시지 : nice to meet you
[Ch.0] 사용자 5의 메시지를 전달합니다.
```

(이어지는 내용...)

```
[Ch.0] 사용자 5 의 메시지 : how are you
[Ch.0] 사용자 5의 메시지를 전달합니다.
[Ch.0] 사용자 5 의 메시지 : quit
[Ch.0] 사용자 5를 채팅방에서 제거합니다.
[MAIN] 채팅방 탈퇴 사용자 탐지 : 5
[Ch.0] 사용자 4 의 메시지 : hello?
[Ch.0] 사용자 4가 혼자여서 메시지를 전달안합니다.
[MAIN] 사용자 5 메시지 : 1
[MAIN] 사용자 5 메시지 : 2 1
[MAIN] 사용자 5가 채팅방 1 에 참여합니다
[Ch.1] 새로운 참가자 : 5
[Ch.1] 사용자 5 의 메시지 : hello
[Ch.1] 사용자 5가 혼자여서 메시지를 전달안합니다.
[Ch.1] 사용자 5 의 메시지 : quit
[Ch.1] 사용자 5를 채팅방에서 제거합니다.
[MAIN] 채팅방 탈퇴 사용자 탐지 : 5
[MAIN] 사용자 5 메시지 : 3
[MAIN] 5 사용자와의 접속을 해제합니다.
```

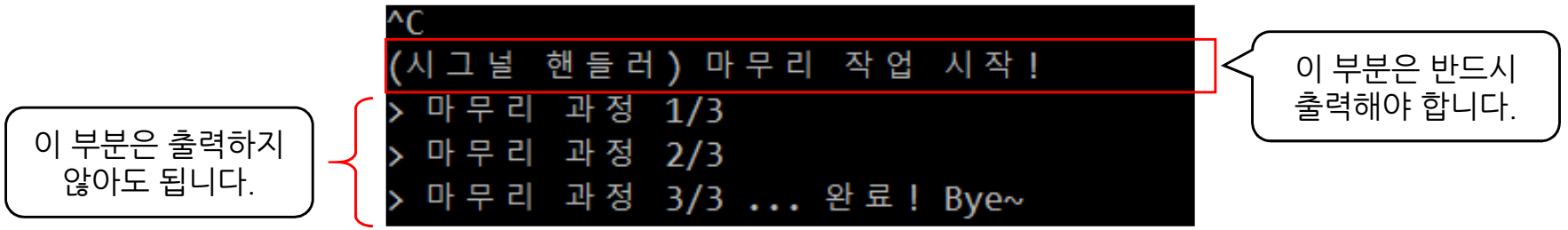
클라이언트가 채팅방에서 탈퇴하는 경우: 해당 채팅방 및 대기실 둘 다 메시지를 출력하거나 또는 둘 중 하나에서만 메시지를 출력해도 됨 (어떤 방식이든 관계 없이, 어떤 사용자가 채팅방을 탈퇴하는지에 대한 로그만 남기면 됨)

[MAIN]은 대기실에서 발생한 이벤트에 대한 로그 메시지 이고, [Ch.0]은 0번 채팅방에서 발생한 이벤트에 대한 로그 메시지를 의미함.

채팅방에 혼자 참여한 상태에서, 채팅 메시지를 입력한 경우

채팅 프로그램, *final*

- 종료 시, 자원 해제
  - 클라이언트는 대기실에서만 접속 종료 기능을 실행할 수 있으며(= 메시지 '3'을 서버로 전송), 메시지 '3'을 서버로 전송한 직후, 클라이언트는 사용하던 모든 자원을 반납하고 프로그램을 종료함
    - 클라이언트가 '3'을 전송하면, 서버는 해당 클라이언트를 위한 소켓을 `close` 함
  - 서버는 무한 반복하는 형태로 구현하고, 서버 터미널에서 `CTRL+C(SIGINT)` 시그널 발생 시 사용하던 모든 자원을 반납하고 프로그램을 종료함



  - 서버/클라이언트 종료 시 해제해야 하는 자원 목록 :
    - 소켓 닫기, 동적 할당 메모리 `free` 시키기, `fork` 한 프로세스 `wait`, 생성한 `thread`를 `join` 하기 등…

채팅 프로그램, *final*

- 채팅방
  - 채팅방은 고유한 ID 값, 방 제목, 참여자/허용인원 정보를 가지고 있음
    - 사용자는 대기실에서 채팅방 목록을 조회할 수 있는데, 이때 아래와 같이 출력되어야 함

```
<MENU>
1.채팅방 목록 보기
2.채팅방 참여하기(사용법: 2 <채팅방 번호>)
3.프로그램 종료
(0을 입력하면 메뉴가 다시 표시됩니다)
```

결과

```
<ChatRoom info>
[ID: 0] Chatroom-0 (1/5)
[ID: 1] Chatroom-1 (0/5)
[ID: 2] Chatroom-2 (0/5)
```

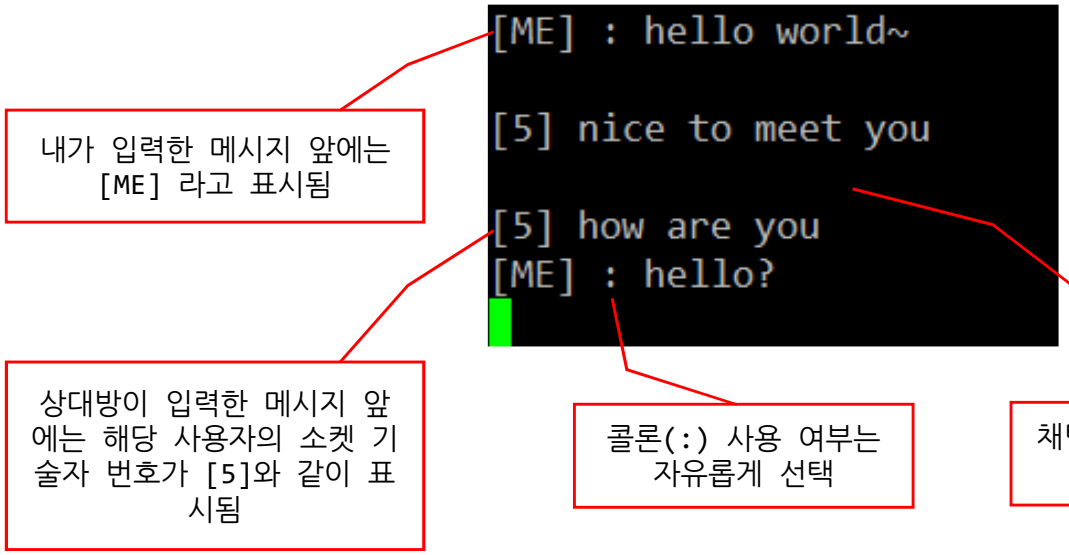
(현재 참가인원 수 / 최대 허용 인원 수) 정보를 출력함

각 채팅방은 방제목을 가지고 있음. 방 제목은 하드코딩 된 문자열을 사용해도 되고, 사용자가 지정할 수 있도록 해도 됨

채팅방의 고유한 ID 값. 사용자는 채팅방에 참여할 때 고유 ID 값을 사용함 (예: 대기실에서 2 0을 입력하면 0번 채팅방에 참여한다는 것임)

채팅 프로그램, *final*

- 가정 (문제를 간소화 하기 위해 아래와 같이 가정함)
  - 서버에 접속할 수 있는 사용자는 최대 15 명 (각 채팅방 최대인원 : 5명, 최대 채팅방 수 : 3개)
    - 최대 인원을 초과해서 접속하는 사용자는 접속을 거부함 (대기실 인원 + 채팅방 인원 최대 수 : 15)
  - 채팅방은 최대 3개까지 개설할 수 있으며, 채팅방을 만들고 없애는 시점은 자유롭게 선택
    - 예를 들어, 서버 프로그램 시작 시 채팅방 3개를 미리 만들 수 있음
    - 채팅방이 비어 있어도 채팅방이 없어지지 않는다고 가정해도 됨
  - 채팅방의 고유번호, 제목은 미리 설정된 번호/제목을 사용해도 되고, 동적으로 변경해도 됨
  - 채팅방에 참여할 수 있는 인원은 최대 5명으로 제한
  - 각 사용자는 ID/별명 등을 설정하지 않음. 사용자의 소켓 기술자 번호를 ID로 사용함



- 왼쪽은 소켓 기술자 번호 4번을 사용하는 클라이언트의 모니터 터미널 화면을 캡처한 것임
- (채팅방에서 소켓 기술자 번호 5번을 사용하는 클라이언트와 채팅 중)
- 여기서 소켓 기술자 번호는, 서버 프로그램에서 accept 이후에 획득한 클라이언트의 소켓 기술자의 번호를 의미함

데모 시나리오

- 시나리오 (최종 데모 시, 아래의 순서대로 실행)
  1. `$make clean` 입력 후, `$make` 입력하여 소스코드 다시 빌드
     - 총 9개의 실행파일이 생성되어야 함
     - 서버 프로그램 실행파일 1개: `server.out`
     - 클라이언트 각각 실행파일 2개(입력 터미널 프로그램 및 메시지 디스플레이 터미널 프로그램) * 4명 사용자

	터미널 프로그램	메시지 디스플레이 프로그램
클라이언트 1	`c1-term.out`	`c1-disp.out`
클라이언트 2	`c2-term.out`	`c2-disp.out`
클라이언트 3	`c3-term.out`	`c3-disp.out`
클라이언트 4	`c4-term.out`	`c4-disp.out`

     - 주의:
       - (필수) 클라이언트 별로 2개의 프로그램을 사용하고, 두 프로그램 간에는 `AF_UNIX` 도메인 소켓을 사용합니다.
       - (필수) 모든 클라이언트 프로그램의 실행파일이 동일한 폴더에 생성되고 동일한 폴더에서 실행될 것이므로, 클라이언트별로 서로 다른 AF_UNIX 주소(경로명+파일명)를 사용하도록 제어하세요.
       - (선택) 컴파일 명령에서 `-D` 옵션을 사용해서, 컴파일 시점에, 각 클라이언트 프로그램이 어떤 주소(경로명+파일명)를 사용할지 결정하도록 코드를 작성하세요.
         - 예) 클라이언트 프로그램 1번 컴파일 시, `-DC1`옵션을 주고, 클라이언트 프로그램 소스코드에서 `#ifdef C1 ...` 구문을 사용하여 클라이언트 프로그램 1번이 내부적으로 사용할 `AF_UNIX` 소켓의 주소(경로명+파일명)를 활성화

데모 시나리오

- 시나리오 (최종 데모 시, 아래의 순서대로 실행)

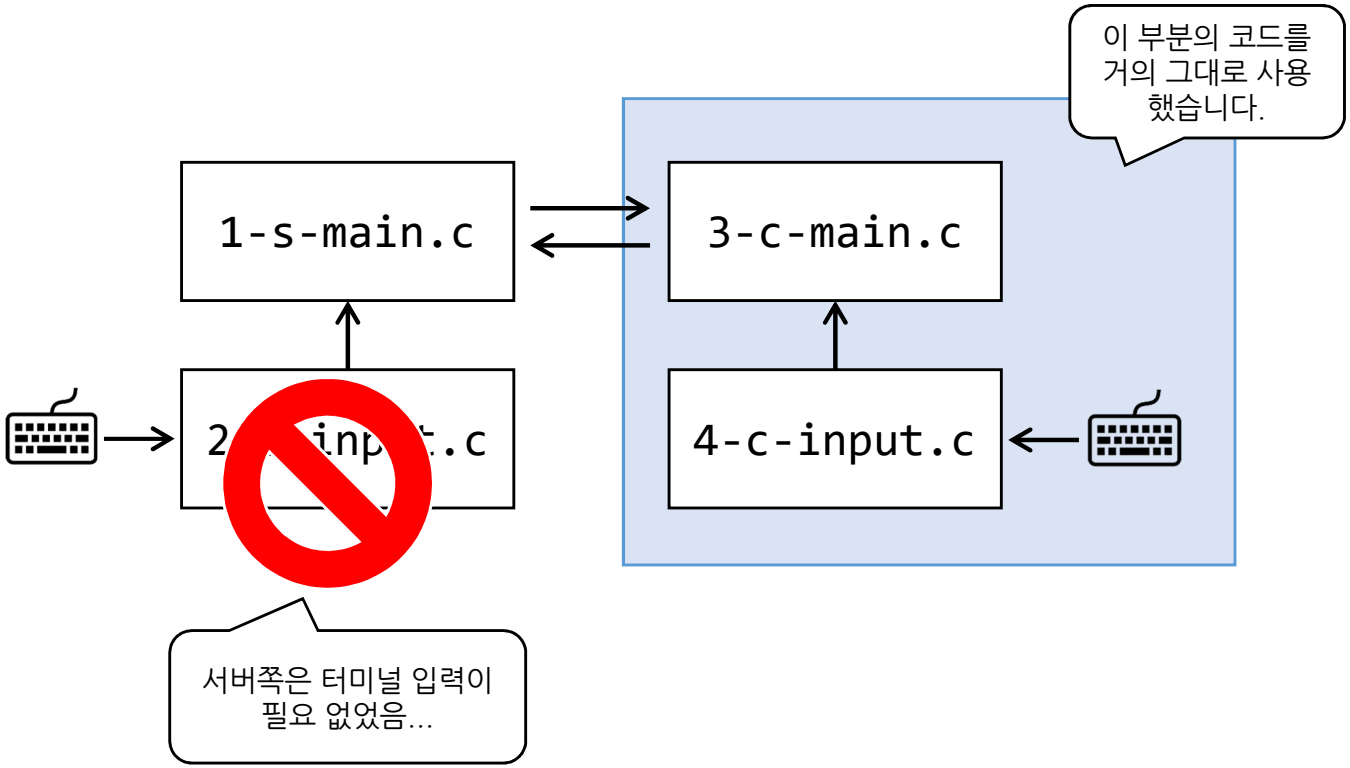
2. 서버 시작 => 클라이언트 1, 2, 3, 4 접속하여 대기실로 이동
3. 클라이언트 1: 채팅방 목록 조회 => 채팅방 0번 참가
4. 클라이언트 2: 채팅방 목록 조회 => 채팅방 0번 참가
5. 클라이언트 1, 2는 동일한 채팅방에서 채팅
6. 클라이언트 3: 채팅방 목록 조회 => 채팅방 1번 참가
7. 클라이언트 4: 채팅방 목록 조회 => 채팅방 1번 참가
8. 클라이언트 3, 4는 동일한 채팅방에서 채팅
9. 클라이언트 3: 채팅방 1번 탈퇴 => 대기실로 이동 => 채팅방 목록 조회 => 채팅방 0번 참가
10. 클라이언트 1, 2, 3는 동일한 채팅방에서 채팅
11. 클라이언트 4: 채팅방 탈퇴 => 대기실로 이동 => 채팅방 목록 조회
12. 모든 클라이언트는 채팅방에서 탈퇴하고, 클라이언트 프로그램을 종료함
13. 서버 터미널에서 [CTRL+C] 입력하여 서버 프로그램을 종료하고, 이 때 서버 프로그램은 시그널 핸들러를 이용해서 종료 절차를 수행해야 함

구현 최소조건

- 소스 코드 구성
  - 서버 프로그램: `server.c`
    - 서버 프로그램을 위한 소스코드는 최소 1개이고, 그 이상의 소스코드를 만들어도 됨
  - 클라이언트 프로그램: `c-term.c, c-disp.c`
    - 각 클라이언트를 위한 소스코드는 최소 2개이고, 그 이상의 소스코드를 만들어도 됨
  - 자동 컴파일: `Makefile`
    - 반드시 `Makefile`을 사용해야 함
- 실행파일 구성
  - `server.c` 를 컴파일 ➔ `server.out` 실행파일 생성
  - `c-term.c, c-disp.c` 를 컴파일 ➔ 총 8개의 실행파일 생성
    - 각 클라이언트별 프로그램은 서로 다른 `AF_UNIX` 주소(경로명+파일명)를 사용해야 함
- 참고
  - 본 설명문에 명시되지 않은 내용은 자유롭게 구현해도 됨

구현 예시

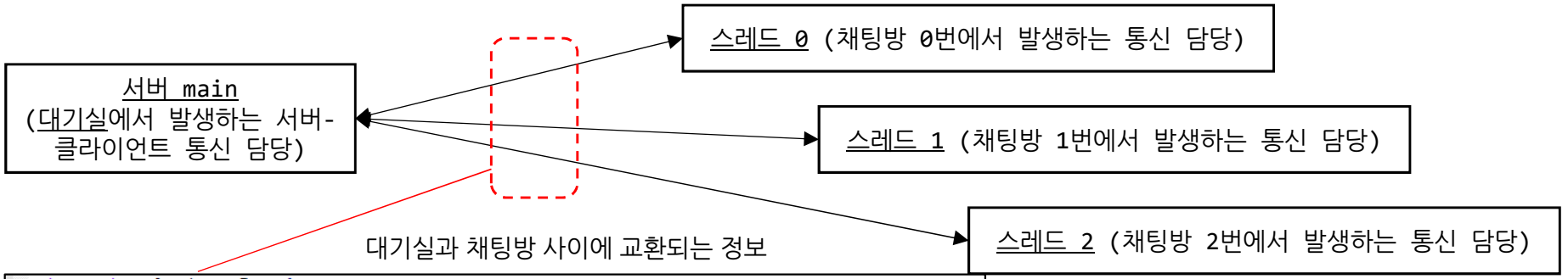
- 클라이언트 프로그램 :
  - 지난 과제에서 구현한 비동기 1:1 채팅 프로그램에서 사용한 3-c-main.c 와 4-c-input.c 프로그램을 거의 그대로 사용



구현 예시

여기 내용은 하나의 예시일 뿐, 본인이 원하는 방식대로 구현하면 됩니다.

- 서버 프로그램 :
  - 프로세스가 아닌 스레드를 사용하여 동시 동작 서버 프로그램 구현
    - 프로세스로 구현하면 프로세스간 정보 교환이 복잡해짐
    - 하지만, 스레드는 메모리 영역을 공유하므로, 스레드간 정보 교환이 쉬운편
    - 교환할 정보는...?
      - 채팅방에 참여하는 클라이언트 소켓 정보를 전달
      - 채팅방에서 탈퇴하는 클라이언트 소켓 정보를 전달 등...
  - 각 스레드는 하나의 채팅방을 전담하여 처리함
    - 클라이언트 한 명에 하나의 서버 스레드를 할당하지 않고, 하나의 채팅방에 하나의 스레드 할당
    - 스레드는 `select` 호출 기반으로 동작
    - 서버 시작 시, 3개의 채팅방을 즉시 생성




```
struct ChatInfo {
    struct ChatRoom room; // 채팅방에 관한 정보 저장을 위한 구조체
    struct UserSockets new_users; // 채팅방에 새로 참여하는 사용자들
    struct UserSockets returned_users; // 채팅방에서 탈퇴한 사용자들
};
```

채팅방 스레드와 대기실이 통신하기 위해 만든 구조체 (클라이언트 소켓 정보 교환 용도)

구현 예시

여기 내용은 하나의 예시일 뿐, 본인이 원하는 방식대로 구현하면 됩니다.

- 서버 프로그램 :

```
main(){
    signal(SIGINT, handler); // 마무리 작업을 수행할 핸들러 함수 등록

    세 개의 스레드 생성: // 세 개의 채팅방을 관리할 스레드를 미리 생성
        pthread_create(..., thread_func, (void*) &chatinfo[i]);
        chatinfo[i] : ID = i인 채팅방의 정보를 관리하기 위한 구조체

    while(TRUE) {
        1. Nonblocking accept 호출
           - 만약, 클라이언트가 connect로 접속 했다면, 메뉴 전송

        2. 채팅 방에서 탈퇴한 사용자가 있는지 확인 (chatinfo[i].returned_users)
           - 있다면, 채팅방에서 대기실로 사용자의 소켓을 이동

        3. (대기실에 있는 사용자만 대상으로) 일정시간 후에 timeout 하는 select 호출
           - readfds 에 변화가 있었다면FD_ISSET 으로 확인하면서 클라이언트의 요청 처리
             (1) 채팅방 정보 요청 -> 채팅방 정보를 사용자에게 전달 (chatinfo[i].new_users)
             (2) 채팅방 참여 -> 해당 채팅방 스레드에 사용자 소켓 정보 전달
             (3) 종료 -> 접속 종료 처리
    }
}

thread[i]_func (chatinfo){
    while(TRUE){
        1. 새로운 채팅방 참가자가 있는지 확인: chatinfo[i].new_users
           - 있다면, 해당 사용자의 소켓 기술자를 chatinfo[i].room로 이동

        2. 채팅방에 참여한 사용자 중에서 메시지를 전송한(send) 사람이 있는지 확인 : select
           - 있다면,
             (1) "quit" 이면, 채팅방에서 탈퇴 처리:
                 해당 사용자의 소켓 기술자를 chatinfo[i].returned_users로 이동
             (2) 그 외 이면, 채팅 메시지 이니까, send 한 사용자 이외의 다른 모든 사용자에게
                 채팅 메시지를 전달하기

    }
}
```

[i]번째 채팅방을 전담할 스레드가 실행할 함수

끝.