서프 게임랜드

조원1 김금영, 조원2 이우영
조원1 소프트웨어학부, 조원2 소프트웨어학부
조원1 ay6656@naver.com, 조원2 ebmmqmf@naver.com

TCP/IP 소켓 프로그래밍 및 멀티쓰레드를 활용한 멀티 게임랜드

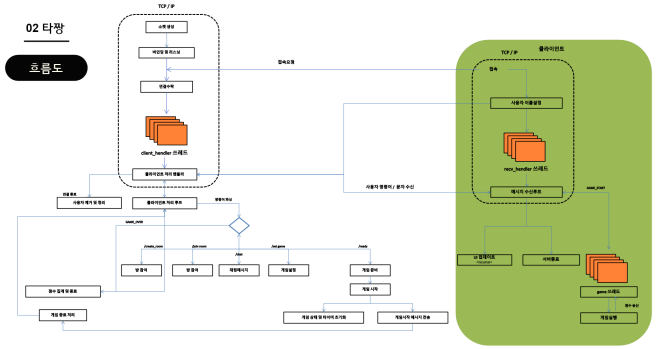
1. Background

수업 및 실습시간에 배웠던 파일입출력, 프로세스, 쓰레드, TCP/IP 소켓프로그래밍을 최대한 활용한 프로젝트가 되어보자! 라는 목표로 다양한 리눅스 함수 및 라이브러리를 이용한 여러 종류의 게임을 만들어보기로 생각하였습니다.

2. Proposed Application

1. 멀티플레이 타자게임 - 타짱 with chat

1.0 응용에 대한 전체 구성도



먼저 게임을 실행하면 클라이언트에서 메인 메인 소켓을 통해 서버에 요청을 보냅니다. 연결이 수락된다면면 사용자는 자신의 이름을 서버로 전송하고, 서버는 해당 이름을 사용자 정보에 등록합니다. 그 후, 스레드를 통해 client_handler 생성하여 각 클라이언트의 연결을 독립적으로 실행합니다.
이렇게 생성된 핸들러는 클라이언트로부터 오는 모든 요청과 명령어를 처리하는 중심 역할을 담당합니다.

그리고 client_handler 스레드를 통해 들어오는 클라이언트 동시 요청 및 공유자원을 뮤텍스를 통해 안전하고 효율적으로 관리합니다.

예를 들어, 사용자가 </chat hello>라는 명령을 보내면 핸들러는 이 명령어를 파싱한 후, 해당 메시지를 같은 방에 있는 다른 클라이언트들에게 브로드캐스트를 시도합니다. 그 후 클라이언트는 recv_handler를 통해 서버로부터 전송된 메시지를 수신하게 됩니다. 예를 들어, 서버가 <game start>라는 메시지를 보내면, 클라이언트의 recv_handler는 이를 확인하고 게임 모듈을 실행합니다.

서버는 게임 시간을 측정하며, 설정된 시간이 되면 클라이언트를 통해 <game over> 메시지를 보냅니다. 이 메시지를 받은 클라이언트는 게임을 종료하고 결과 점수를 송수신하여 최종 결과를 출력한 후 종료합니다.

1.1 어떤것을 이용했는지?
파일 입출력 (File IO):
서버 클라이언트에서 출력 에러 및 문제점을 파악하기 위해 로그파일 생성 및 메시지 기록합니다.
파이프 및 쓰레드 (Pipe & threads):
서버 - 각 클라이언트 연결을 독립적으로 처리하기 위해서 pthread를 이용하여 client_handler 생성 pthread_mutex를 이용하여 공유자원을 보호하였습니다.
클라이언트 - 서버로 부터 메시지 수신 및 게임실행을 위해 pthread를 사용하여 recv_handler를 생성하고. pthread_mutex를 사용하여 공유자원 및 ncureses UI 창 접근을 관리 하였습니다.
소켓(Socket):
TCP/IP 소켓을 사용하여 클라이언트와 연결 수락 및 관리 및 서버 클라이언트간 메시지 송수신 하였습니다.

1.2 OSS 사용 여부

1) 오픈소스 Linux UI인 ncurese를 사용하여 게임창 UI를 구현

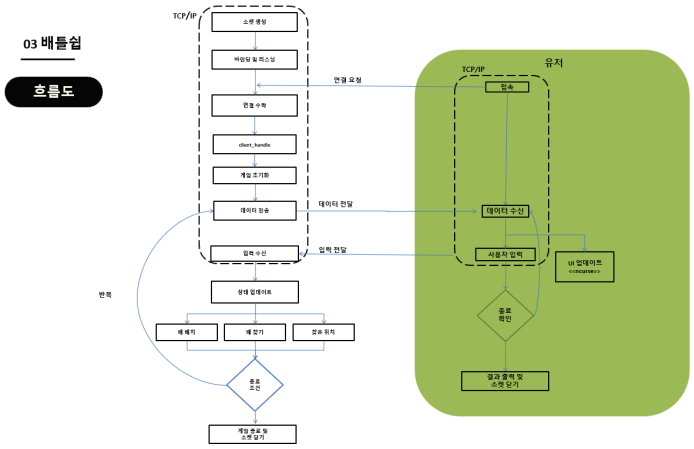
2) 산성비 게임의 단어 체크 부분에 대해 어려움을 오픈소스인 github : typing_game (github.com/YoonShinWoong/typingGame) 자료의 word_check 함수 부분을 참고하여 단어체크 기능을 game_word_check 함수로 구현하였습니다.

1.3 응용의 강점 및 강조 부분

1) FILE I/O를 이용한 로그파일 기록 및 에러 처리
   프로젝트를 진행하며 예상치 못한 에러 및 출력 오류를 겪었는데 이를 방지 및 에러의 원인을 자세하게 파악하기 위해 출력 및 중요 point마다 로그를 파일입출력을 통해 server.log, client.log 으로의 기록을 통해 에러 및 원인을 쉽게 파악할수 있었습니다.
2) 멀티 쓰레드를 이용한 클라이언트 요청 처리
   client_hendler에서 thread를 이용하여 각각의 클라이언트들의 요청을 독립적으로 처리 및 사용자의 공유자원에 대한 요청 및 접근은 pthread_mutex를 이용하여 공유자원 동시접근을 방지 했습니다.
3) TCP/IP 소켓을 이용한 메세지 및 명령어 수행
   TCP/IP 소켓을 이용하여 클라이언트 서버간 메시지 송수신 및 파싱된 명령어에 추출 및 지정된 작업을 수행하는 시스템을 구현하였습니다.

2. 배틀쉽

2.0 응용에 대한 전체 구성도



배틀쉽은 단일 프로세스 기반의 서버 클라이언트 동작을 수행합니다. client_handler를 통해 각 클라이언트들의 보드배치 데이터 및 공격 좌표, 현재 공격 차례 파악 정보를 송수신하며 게임을 진행합니다.

클라이언트에서 공격할 좌표를 선택하고 해당좌표를 서버로 수신하고나면, 서버는 입력받은 좌표를 통해 HIT, MISS판단을 합니다. 그 후 클라이언트에게 공격수행 결과를 송신합니다.

2.1 어떤것을 이용했는지?

파일 입출력 (File IO):

서버 클라이언트에서 출력 에러 및 문제점을 파악하기 위해 로그파일 생성 및 메시지 기록합니다.

소켓(socket):

TCP/IP를 이용하여 각 클라이언트들의 보드 배치 및 공격 좌표를 송수신 및 HIT/MISS 파악합니다.

2.2 OSS 사용 여부

1) 오픈소스 linux UI인 ncurese를 사용하여 게임창 UI를구현

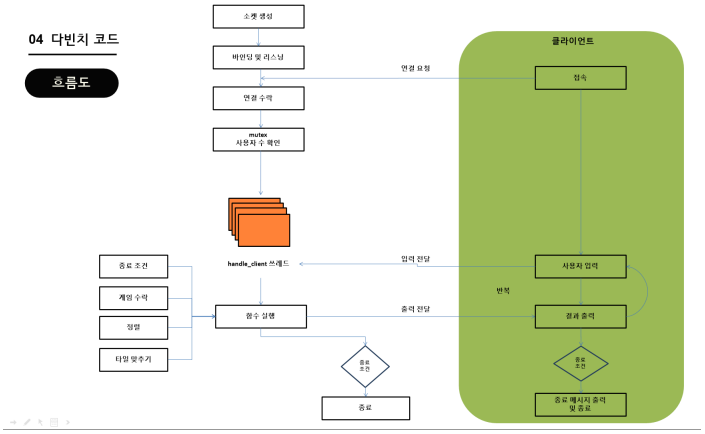
2) 멀티플레이에서 배치 갱신, HIT/MISS 및 게임종료 처리에 대해 파악하기 위하여 github: BattleshipClientServer (github.com/alexguillon/BattleshipClientServer) 의 getclient,play함수를 참고 하였습니다.

2.3 응용의 강점 및 강조 부분

1) 소켓을 이용한 그리드 배치 갱신 및 HIT/MISS처리
   배틀쉽 멀티게임에서 상대방의 공격에대한 HIT/MISS처리 및 공격 후 배치 그리드 갱신은 멀티게임을 구현하며 가장 어려운 부분이었습니다. 클라이언트에서 write()통하여 공격좌표를 송신하면 서버에서 좌표를 확인후 상대 그리드의 HIT/MISS를 판단 및 결과 메시지를 송신합니다. 그리고 클라이언트에서 HIT/MISS 메시지에 따라 보드에서 공격 및 방어 표시를 갱신하도록 구현하였습니다.

3. 다빈치 코드 - Coda

3.0 응용에 대한 전체 구성도



서버에서 먼저 소켓을 생성하고 바인딩 후 리스닝 상태에 들어갑니다. 클라이언트의 접속이 요청되면 접속을 수락하는 동시에 mutex를 통해 사용자 수를 관리하고, 접속된 사용자 수를 확인합니다. 사용자 수가 일정 수에 도달하면 게임을 시작하고, 서버와 클라이언트 간에 정보 전달이 이루어집니다. 클라이언트에서 사용자 입력이 주어지면 서버는 그 입력을 받고 처리한 후에 결과를 클라이언트에 보냅니다. 이와 같은 행동을 반복하다가 종료 조건에 맞게되면 게임을 종료합니다.

3.1 어떤 것들을 응용하였는지?

1) 서버와 클라이언트 간 데이터 전송을 원활하게 하기 위해 TCP/IP를 사용하여 게임 상태를 제어하였습니다.
2) mutex(뮤텍스)를 사용하여 다중 클라이언트가 동시에 접속했을 때 사용자 수를 안전하게 확인하고, 각 사용자 턴이 올바르게 진행될 수 있도록 턴을 관리해서서 경쟁 조건을 방지합니다.

3.2 OSS 사용 여부

1) 오픈소스 linux UI인 ncurese를 사용하여 게임창 UI를 구현하였습니다.

3.3응용의 강점 및 강조 부분

1) 안정적인 데이터 전송
   TCP/IP 프로토콜을 사용하여 서버와 클라이언트 간 데이터 손실 없이 안정적으로 송수신할 수 있습니다.
2) 경쟁 조건 방지 및 동기화
   Mutex를 활용하여 다중 클라이언트 환경에서 사용자 수 확인 및 게임 턴 관리를 안전하게 수행합니다.
3) 직관적인 게임 UI
   오픈소스 Linux UI 라이브러리인 ncurses를 활용하여 텍스트 기반 게임 화면을 구현하였습니다.

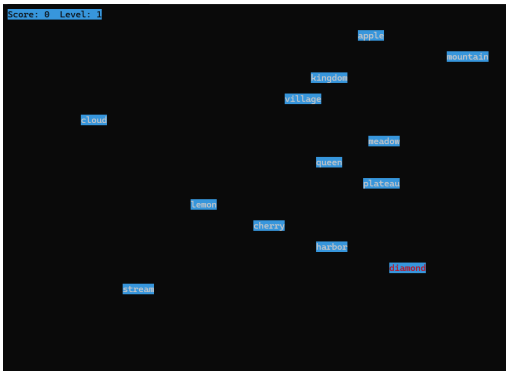
3. Usage Checklist

내용	O / X
Linux intro (week 2)	O
Dir/File (week 4)	X
File IO (week 5)	O
Process (weeks 7 & 9)	O
Pipe & threads (week 10)	O
Socket (weeks 11 & 12)	O

4. Demonstration

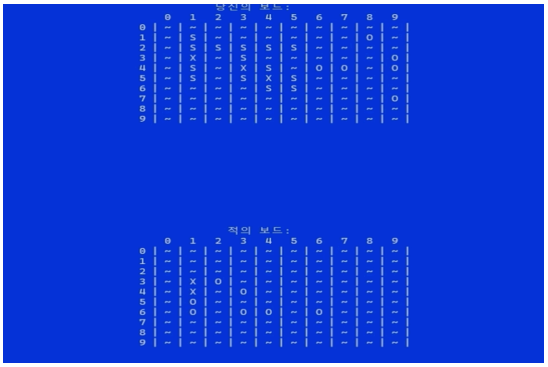
1. 멀티플레이 타자게임 - 타짱 with chat

게임접속시 채팅창이 등장합니다.
여기에서 다양한 명령어를 통해 채팅, 방생성, 방리스트 확인 ,모드확인, 게임시작을 할 수 있습니다.

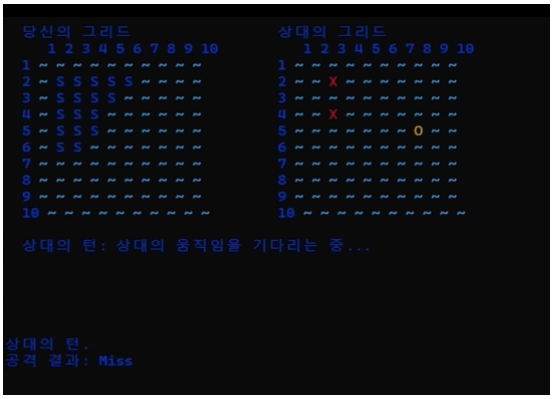


게임 화면 - 소나기 게임으로 90초간 진행되어지며 한 유저가 목표점수(150점) 에 도달하면 게임에 승리하여 승리자 출력이 되어지고, 게임종료 되어집니다.

2. 배틀쉽

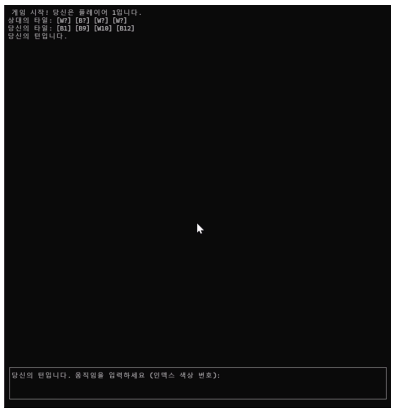


싱글플레이 모드 : 싱글 유저가 다양한 난이도 (easy, nomal, hard) 모드를 통해 컴퓨터와 배틀쉽 게임을 진행할 수 있습니다.



멀티플레이 모드 (배틀넷) : 2명의 유저가 해당 ip와 포트를 입력 후 차례대로 공격을 및 공격 상대방 보드의 성공여부를 확인하며 멀티게임이 진행되어집니다.

3. 다빈치 코드 게임



TCP/IP 접속 이후 상대와 자신의 타일을 나누고, mutex를 통해 한턴에 한명의 플레이어만 추리를 하도록 턴을 제어하였습니다.

5. Conclusions

보완사항

·멀티플레이 타자게임 - 타짱 with chat

초기 설계에서는 exec함수군을 활용하기로 계획을했고, 유저가 게임준비가 되었다면 execl함수를 사용하여 게임실행하려고 하였으나 커서 충돌로 인하여 구현하지 못하였습니다.

게임모드 설정 명령어 파싱부분에서 서버 클라이언트 통신이 불안정 모든 유저가 방에서 퇴장한 경우에도 방이 존재하는 경우에 대한 처리를 구현하지 못했습니다.

· 배틀쉽

멀티 플레이 게임에서 ncurese 인터페이스 충돌로 인한 커서 사라짐 현상 개선 필요합니다.

· 다빈치 코드 게임

3~4명까지 client가 접속할 수 있지만, 상대 타일을 보여주는 과정 때문에 2명만 이용할 수 있습니다.

남은 타일이 없는 상태에서 추리를 실패하면 자신의 패가 랜덤 오픈되어지는 규칙에 대해 프로그램상 추가적인 구현이 필요하지만 현재 구현한 코드에서는 2인 게임이므로 해당 기능을 구현하지 못했습니다.

느낀점

김금영

•프로젝트를 진행하며 수업에서 배웠던 간단한 소켓 프로그래밍을 통해 게임을 구현 가능할 것 이라고 생각했지만 구현 과정에서 동기화와 같은 문제점들을 겪었고, 해당 문제들을 해결하며 역량을 키울 수 있었던 프로젝트였습니다.

이우영

•이번 프로젝트를 진행하면서 작은 기능을 구현하는데 너무 오랜시간이 걸려서 아쉬운 점이 크게 남아있었습니다.

소켓 프로그래밍을 구현하면서 관련 지식을 키울 수 있어서 좋은 프로젝트라고 생각했습니다