**UNIX시스템프로그래밍 설계과제**

설계과제 명 : 클라이언트와 서버 통신

제출날짜 :

팀명 :

팀원 : 2020150035 이주영(팀장), 2020150040 정승민

1. 문제 정의 및 요구 사항 분석

* 문제 정의 및 설계 과제 목표
  + 문제 정의
    - UNIX 환경에서 프로세스 통신 기법과 멀티스레딩을 활용하여 서버와 2명의 클라이언트로 틱택토 게임을 설계하는 것
  + 설계 과제 목표
    - 이를 위해 서버와 클라이언트를 파이프 시스템으로 연결, 멀티스레딩을 구현하되 synchronization tool을 이용하여 동시성을 제어 해야함
* 요구사항 분석
  + 틱택토 게임 매커니즘
    - 3 x 3 의 격자판에 두 명의 플레이어가 한 차례씩 말을 둬서 게임을 끝날 떄까지는 두는 것, 게임이 끝나기 위해서는 어느 줄(가로,세로대각선)이든지 한 줄이 모두 동일한 말로 통일 되면 게임 승리 혹은 더 이상 말을 둘 수 없을 때 게임 무승부
  + 시스템 요구 로직
    - 멀티스레딩을 하되 동시성 및 데이터 무결성을 제어 및 유지
    - IPC 혹은 파이프 를 통한 통신 프로세스간 통신 기법
    - 프로세스 시간 측정(성능 측정을 위한 것)

1. 시스템 설계

* 소프트웨어 설계도
  + Shared memory

그림

* + 파이프

도표, 텍스트, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

* 쓰레드 설계
  + Shared memory
    - 서버
      * + 게임 관리자 스레드: 게임 상태 관리, 턴 전환, 승리 조건 확인
        + 디스플레이 스레드: 현재 게임 보드 상태를 주기적으로 출력
        + 클라이언트 핸들러 스레드: 클라이언트의 차례를 관리하고 입력을 처리
    - 클라이언트
      * 입력 스레드: 사용자 입력 처리 및 보드 업데이트
      * 업데이트 스레드: 상대방의 움직임 확인 및 화면 갱신
  + 파이프
    - 서버
      * 메인 + 게임 모니터링 + 클라이언트(2개)
        + 메인 : 기본 동작을 구현 및 스레드 총괄, 클라이언트 연결
        + 게임 모니터링으로 게임 종료를 병렬로 확인
        + 연결된 클라이언트 2개의 통신을 담당, Semaphore로 통제
    - 클라이언트
      * 메인 + 수신 + 입력 + 상태 모니터링
        + 메인 : 메인 스레드에서 총괄
        + 수신 : 서버로부터 메시지 수신 및 처리
        + 입력 : 사용자로부터 입력을 처리하고 서버로 전송
        + 상태 : 게임 종료 플래그를 주기적 확인, 스레드 종료
* 공유 자원(또는 변수) 정의와 동기화 기법의 설계
  + 파이프
    - Mutex로 구조체와 파일접근에 대한 접근을 동기화 및 클라이언트의 차례가 오면 입력 스레드를 깨우기 위한 용도로도 쓰임 ( 아래는 일부)
    - Semaphore로 각 클라이언트의 턴을 제어
  + Shared memory
    - pthread\_mutex\_t 로 공유 자원 접근을 안전하게 관리
    - pthread\_cond\_t 특정 조건에서 스레드를 대기하거나 깨우기

1. 시스템 구현 및 데모
   * 소스코드 중요 부분(shared memory)
     + 서버
       - 공유 메모리 초기화 및 동기화 설정(main)

뮤텍스와 조건 변수 데이터 일관성을 보장하고 스레드 간 동작을 동기화 하기 위한 코드들

* + - * game\_manager\_thread

게임의 종료 조건을 확인하고 조건이 충족되면 게임이 종료되도록 함

* + - * client\_handler\_thread

각 클라이언트가 자신의 차례를 기다리고 입력을 처리하도록 하는 역할

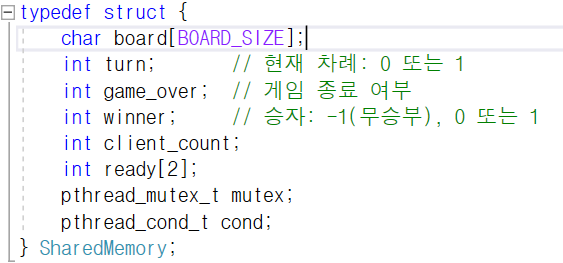
* + - 클라이언트
      * input\_thread

사용자가 올바른 위치를 입력하는지 검증하고 틱택토 보드에 해당 위치를 업데이트, 유요하지 않은 입력 처리 및 턴 전환 로직이 포함됨

* + - * update\_thread

틱택토 보드 상태를 지속적으로 최신 상태로 표시함

* + - 사용 구조체



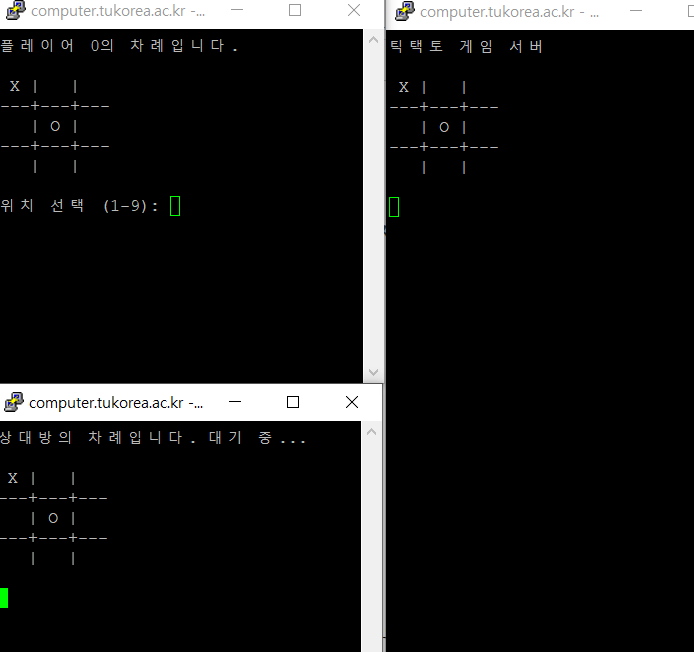
* + - 데모 시나리오

클라이언트1이 먼저 위치 선택

클라이언트2가 다음 선택

게임이 종료된다면 승부여부를 파악하여 출력 이후 종료

* + - 프로그램 실행결과





각 클라이언트가 순서에 맞게 입력을 받으며 받은 값이 서버, 클라이언트에 정상적으로 업데이트 되는 것을 볼 수 있다.

* 소스코드의 중요 부분(파이프)
  + 로직
    - 서버
      * 게임 초기화 후 두 클라이언트 연결 대기
      * 각 클라이언트 연결 후 스레드 및 파이프 생성 및 연결, 시작
      * 게임 종료 전까지 게임 수행 후 종료 조건이 확인 시 종료
      * 종료 후 시간 및 결과 알려줌
    - 클라이언트
      * 서버와의 통신 채널 생성, 필요한 스레드 생성 및 병행 처리
      * 수신 스레드가 알림 받아서 클라이언트 게임 수행을 입력 스레드로 함
  + 공유 자원에 대한 동기화 기법
    - 게임 정보 구조체

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명 

* + - * 예시

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

구조체 데이터 접근에 대해 뮤텍스를 사용하여 제어함

* + - 클라이언트 정보

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

* + - * 위 같이 선언하여 클라이언트의 정보와 세마포어로 턴제 동기화 텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

        자동 생성된 설명
* 데모 시나리오

|  |  |
| --- | --- |
| 서버 | 클라 |
| Waiting for clients to connect...  Client 0 connected.  Client 1 connected.  Game monitor thread started.  Client handler for Player 0 started.  Client handler for Player 1 started.  Player 0's turn.  ...  Player 0 wins by row 0!  Game monitor thread detected game over.  Game ended.  Result: Player 0 wins!  Total Runtime: 30.123 seconds  Total Input Time: 12.456 seconds  Adjusted Time (Runtime - Input Time): 17.667 seconds  Server exiting. | Connected to server as Player 0.  Listener thread started for Player 0.  Input handler thread started for Player 0.  Status monitor thread started for Player 0.  === Game Board ===  Turn:0  | |  | |  | |  Received 'Your Turn' from server.  Enter your move (row and column): 0 0  Input Time: 2.345 seconds |

* 프로그램 실행 결과
  + 파이프

텍스트, 스크린샷, 메뉴이(가) 표시된 사진

자동 생성된 설명

1. 현실적 제한 요소 및 성능 측정

* 한계
  + 쓸데 없이 복잡한 세마포어랑 뮤텍스를 혼용하여 사용, 파일이란 중간 과정을 끼고 하니 좀 느린 거 같다.
* IPC 기법의 성능 분석

1. 팀워크
   * 팀의 작업
     + 문제 정의 및 요구 사항 분석을 같이 토의로 진행
     + 각각 기법을 나눠서 프로그램 제작
   * 역할
     + 이주영(팀장): shared memory 로 틱택토 게임 구현, 보고서 제작
     + 정승민: pipe 로 틱택토 게임 구현, ppt, 발표
2. 결론 및 총평
   * 결론
     + 파이프로 작성하여도 순차적 실행하면 오래 걸릴 것을 스레드로 병렬적 실행을 하다 보니 더 빠르게 처리가 가능한 것으로 보인다.
   * 총평
     + 파이프로 작성하다 보니 뭔가 좀 더 쉬운 길이 있는 데 불편하게 돌아가는 느낌이었다. 더구나 초기 구상에는 가위바위보로 순서 정하기, 스레드를 늘려 방을 만들 생각을 하였으나 작성하다 보니 오만인 것을 깨 닳았다.
   * AI 사용 부분
     + 정승민
       - 게임 종료 플래그가 종료되지 않음에 문제가 생겨서 이에 대해 OpenAI gpt o1-preview, o1-mini 를 사용하여 해결하였다.
     + 이주영
       - Shared memory에 쓰일 구조체 형태 제작, 승자가 누구인지 확인하는 함수, 클라이언트 헨들러 함수를 gpt o1-mini 를 사용하여 해결하였다.