알고리즘 겨루기 프로젝트 보고서

202010891 컴퓨터과학과 이근희

202010848 컴퓨터과학과 김성민

1. 서론

학생들 대부분은 알고리즘을 공부할 때 백준 등의 채점사이트를 이용한다. 채점사이트에서 알고리즘을 공부할 때 학생들은 문제를 풀고 채점 받고, 풀고 채점 받는 단순한 과정을 반복하게 된다. 우리 팀은 이러한 단순 반복속에서 지루함을 느낀 학생들과 자신의 실력을 친구와 비교해보고 싶은 학생들이 있을 것이라 생각했다.

이에 알고리즘 겨루기 프로젝트는 기존의 단순한 알고리즘 공부 방법에서 벗어난 놀이적인 알고리즘 공부 방법 제공과 함께 친구와 실력을 겨룰 수 있는 장을 제공하는 것을 목표로 고안되었다.

1. 주제 설명

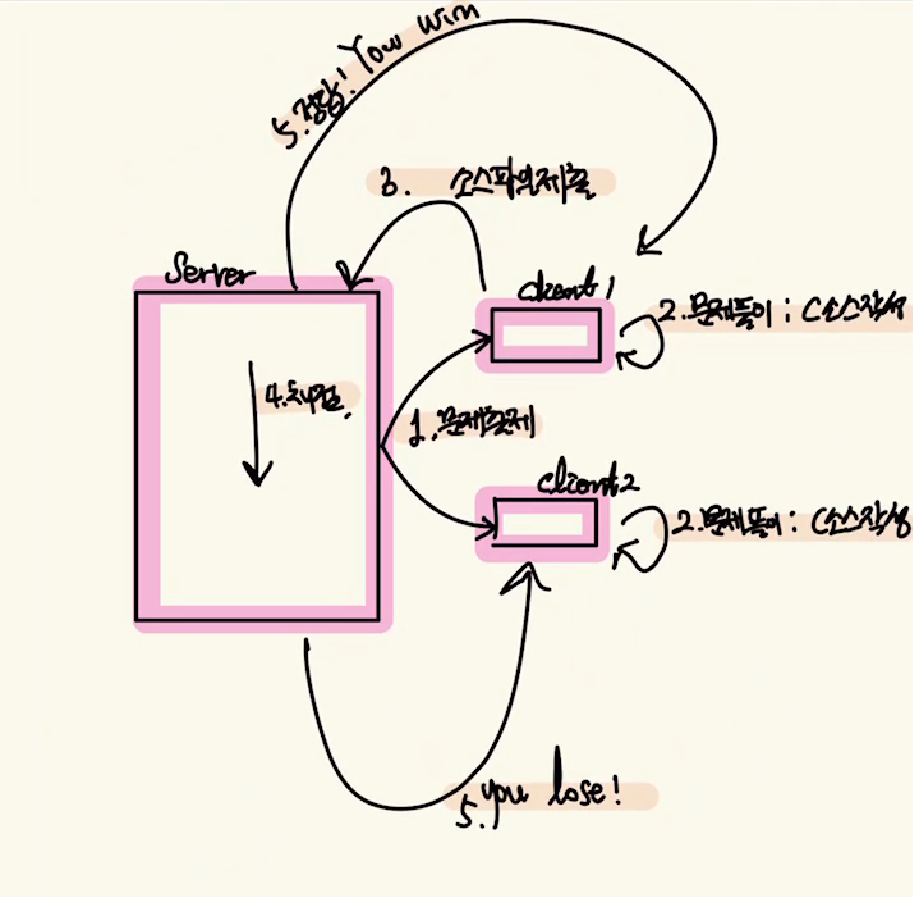
알고리즘 겨루기 프로그램은 랜덤으로 출제된 알고리즘 문제를 누가 더 빨리 푸는 지 2명의 클라이언트가 겨룰 수 있도록 하는 프로그램이다.

알고리즘 겨루기 프로그램은 1:1겨루기만을 지원한다. 따라서 서버 하나에 클라이언트 둘이 매달리는 구조가 되며, 각각의 클라이언트는 서버의 쓰레드가 핸들링 한다.

서버에 클라이언트 둘이 모두 연결되어 겨루기가 시작되면 참여자들은 소켓 통신을 통해 서버로부터 랜덤 문제를 받게 된다. 이후 참여자들은 출제된 문제를 c소스로 편집하여 서버에 제출, 상대보다 빠르게 서버에게서 정답임을 인정받음으로써 겨루기에 승리할 수 있다.

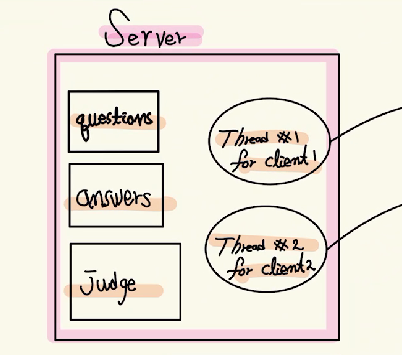
1. 프로그램의 전체적인 흐름

시스템의 전체적인 흐름을 먼저 소개하겠다. 각 부분의 세부적 구현 내용은 이어지는 시스템 설계도 목차에서 설명할 것이다.



1. 소켓 연결: 서버와 두 클라이언트가 소켓으로 연결 된다. (같은 공유기를 사용하는 네트워크 환경에서 연결 될 수 있음)
2. 문제 출제: 서버가 자신이 가지고 있는 문제들 중 랜덤으로 문제를 골라 client들에게 출제한다. (두 클라이언트는 같은 문제를 두고 겨루게 됨)
3. 문제 풀이: 클라이언트들은 출제된 문제를 c소스로 편집하며 푼다.
4. 소스파일 제출: 클라이언트가 자신이 정답이라고 생각하는 소스파일을 서버에 제출한다.
5. 서버는 제출 받은 제출물을 본인의 채점프로세스로 채점해본다.
6. 정답여부를 확인하고 승자가 나왔다면 이를 클라이언트들에게 알린다. (오답이면 오답임을 알리고 다시 제출물을 기다린다.)

1. 시스템 설계도와 사용된 기술



먼저 서버의 설계도에 대해 설명하겠다. 그림처럼 서버는 문제목록(questions)과 답지 목록(answers)을 가지고 있으며 채점하는 심판인 judge를 구성요소로써 가진다. 서버와 클라이언트의 연결은 소켓을 매개로 되어 있으며 서버는 연결된 두명의 클라이언트를 각각 쓰레드로 핸들링 한다.

* 쓰레드 구조의 서버

서버는 소켓을 통해 클라이언트와 통신하는데 클라이언트마다 쓰레드 하나를 할당하여 각 클라이언트를 handling한다. 다만 서버는 클라이언트 2명이 연결되면 추가적인 연결을 거부하며 두 클라이언트의 1:1겨루기에만 집중한다. (서버에 두 클라이언트만 매달릴 수 있음)따라서 서버는 현재 몇 명의 클라이언트가 연결되었는지 전역변수로 관리하고 있으며 여러 쓰레드가 동시에 이 값을 변경하는 것을 막기 위하여 mutex를 사용, critical section에 들어갈 때 마다 mutual exclusion을 보장하도록 설계되어 있다.

cf> server.c critical section

텍스트이(가) 표시된 사진

자동 생성된 설명

cf> server.c 두 클라이언트만 연결되도록 설정 하는 코드

텍스트이(가) 표시된 사진

자동 생성된 설명

cf> server.c 연결된 클라이언트에게 쓰레드 할당 및 임계영역 변수 변경

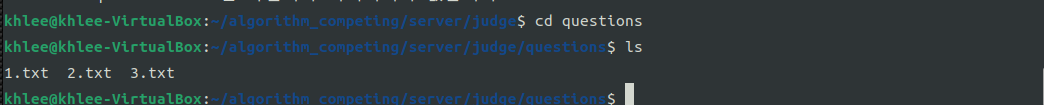
텍스트이(가) 표시된 사진

자동 생성된 설명

* 서버의 문제 목록과 답지 목록

서버는 디렉토리 안에 문제들을 파일로써 관리하며 해당 문제에 대한 답지들도 디렉토리 구조 속 파일로써 관리한다. 서버가 랜덤으로 문제를 출제하면 채점기는 출제된 문제에 대한 답지를 바탕으로 채점하게 된다. 채점기는 예시입력을 넣었을 때 채점 대상이 모범출력과 같은 값을 출력하는지 5가지의 사례를 확인하는 식으로 작동하기에 답지 파일은 예시 입력과 예시입력에 대한 답안을 저장하고 있다.

cf> 문제 파일들



cf> 답지 파일 3.sol

텍스트이(가) 표시된 사진

자동 생성된 설명

3번 문제를 채점 받는 프로그램은

18을 입력했을 떄 4를 출력하고

4를 입력했을 때 -1을 출력하고

6을 입력했을 때 2를 출력하고

9를 입력했을 때 3을 출력하고

11을 입력했을 때 3을 출력해야 정답으로 인정받을 수 있다.

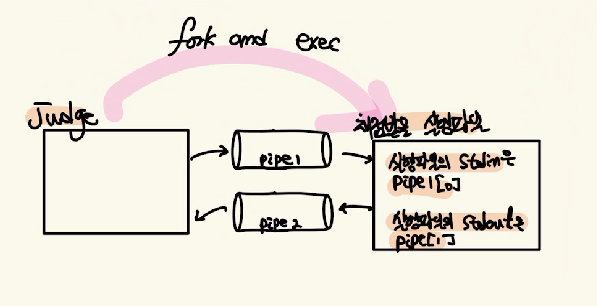
* 서버의 judge(채점기)

채점기는 서버 프로그램에서 내부적으로 사용하는 모듈화 된 프로그램이며 명령행 인자들과 함께 다음과 같이 독립적으로 실행될 수도 있다.

./judgeproc [실행파일] [답지파일]

채점 방식은 위에서 언급했듯 답지 파일의 5가지 사례에 대해서 올바른 출력을 내는지를 확인하는 방식이다. 이에 채점기는 fork를 이용 채점 받을 실행파일을 실행시키고 입력을 넘겨 출력 값을 관찰하는 구조를 가진다.

다만 대부분의(거의 모든) 알고리즘 문제에 대한 풀이가 그렇듯 채점을 기다리는 제출물은 stdin으로부터 입력을 기다리고 있을 것이며 출력 또한 stdout으로 하게 될 것이다. 따라서 채점기는 fork한 실행파일의 stdin으로 입력 예시를 주입해야 하고 실행파일의 stdout을 관찰해야 하기에 양방향 pipe와 입출력 redirection을 사용했다. 해당 구조는 아래의 그림과 같다.



Judge는 실행파일을 exec하기 위해 fork를 사용하지만 judge또한 서버의 fork로 인하여 불리게 된다. 서버가 클라이언트의 소스코드 제출을 감지했다면 system()함수를 이용해 유저가 제출한 소스코드를 빌드하고 fork를 이용해 judgeproc을 실행시키는 것이다. 따라서 서버가 fork하여 judge를 부르고, judge가 fork하여 실행파일을 실행시키는 삼중구조로 채점이 이루어 진다.

cf> 멀티 쓰레드 환경에서의 fork가 문제될 수 있다는 문헌이 많아 자세히 읽어보았고 해당 쓰레드 구조를 유지하며 fork할 필요가 없는 우리는(채점기만 불러주면 끝이니까) 문제 없이 fork를 사용할 수 있었음을 알린다.

cf> judgeproc.c 파이프 구조와 fork후에 exec되는 실행파일의 입출력 redirection

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

cf> judgeproc.c 실행파일 exec하는 부분

텍스트이(가) 표시된 사진

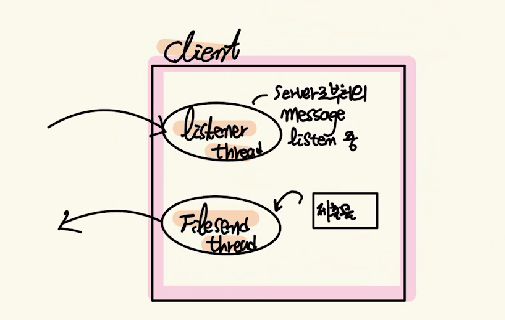
자동 생성된 설명

cf> server.c에서 judgeproc을 호출하고 반환값을 체크하는 부분

텍스트이(가) 표시된 사진

자동 생성된 설명

이어 클라이언트의 구조에 대해 설명하겠다. 클라이언트는 두개의 쓰레드를 가진다. 하나는 서버로부터 오는 메시지를 수신하는 쓰레드이며 다른 하나는 소스 코드를 서버로 작성하기 위한 쓰레드 이다. 수신하는 쓰레드와 파일 전송하는 스레드 모두 별도의 라이브러리가 아닌 소켓 프로그래밍을 기반으로 동작한다. (ftp를 쓰지 않고 소켓 프로그래밍을 응용한 형태로 사용했다.)



* Listener Thread

Listener는 소켓 또한 파일이라는 개념을 응용하여 만들어졌다. (read를 이용 소소켓으로부터 읽어온다) 무한 반복으로 서버와 연결되어 있는 소켓을 쳐다보며 서버가 보낸 메시지가 있다면 받아와서 화면에 뿌려준다. 다만 승패가 결정되어 게임이 종료되게 되면 무한반복을 탈출하여 쓰레드가 종료되어야 함으로 if문을 사용, 서버에게서 게임 종료 메시지를 수신하면 제어를 무한 반복문의 바깥으로 옮겨 쓰레드가 소멸되도록 유도하는 코드를 사용했다.

cf> client.c 리스너 쓰레드의 핵심 동작

텍스트이(가) 표시된 사진

자동 생성된 설명

* Send file thread

클라이언트는 소스 코드를 완성 시켰다면 서버에 제출해 채점을 받아야 한다. 이 때 서버에 파일을 전송하는 역할을 send file 쓰레드가 하는데 소켓 기술자에 저수준 쓰기를 하는 방식으로 파일을 전송한다.

cf> client.c 파일 전송 쓰레드 핵심 동작

텍스트이(가) 표시된 사진

자동 생성된 설명

1. 사용 매뉴얼

* 같은 공유기 내의 소켓 통신만이 가능합니다. 따라서 ubuntu 설정을 다음과 같이 변경합니다.
* VM 우분투 설정 -> 네트워크 -> 어댑터에 브리지텍스트이(가) 표시된 사진

  자동 생성된 설명
* 서버 컴퓨터 할 일
  + 압축 파일 해제 -> server 디렉토리로 이동
  + server디렉토리의 server.c에서 매크로를 자신의 ip로 수정



* + gcc -o server server.c 를 이용해 컴파일한다. (혹 라이브러리를 인식 못해 컴파일이 되지 않는다면 -lpthread를 붙여 다시 컴파일 하면 됨)
  + server디렉토리 안에 있는 judge 디렉토리로 이동, gcc -o judgeproc judgeproc.c를 이용해 judgeproc.c를 컴파일 한다.
  + ./server 를 이용해 서버 실행
* 참가자들이 할 일
  + 압축 파일 해제 -> client 디렉토리로 이동
  + client디렉토리의 client.c에서 매크로를 서버의 ip로 수정



* + gcc -o client client.c를 이용해 컴파일 한다. (혹 라이브러리를 인식 못해 컴파일이 되지 않는다면 -lpthread를 붙여 다시 컴파일 하면 됨)
  + 서버가 실행된 상태에서 ./client를 이용해 실행

이후 알고리즘 겨루기에 참여하는 클라이언트들은 서버에서 보내주는 안내에 따라 주면 된다.

1. 사용 예 캡처는 생략 녹화 영상으로 대체

끝 ..