**System Programming Project 2**

담당 교수 : 김영재 교수님

이름 : 이건영

학번 : 20181662

1. **개발 목표**

여러 client의 동시 접속 및 서비스가 가능한 concurrent 주식 서버를 Event 기반, Thread 기반의 두 가지 방식으로 구축하고 그 성능을 평가한다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Task 1: Event-driven Approach

Server의 단일 프로세스 내에서 connfd들을 array에 관리한다. Select 함수를 사용해서 pending input이 있는 지 확인한 후, 만약 있다면 event로 간주하여 해당 input에 맞는 기능을 수행한다.

1. Task 2: Thread-based Approach
2. Task 3: Performance Evaluation
   1. **개발 내용**

* **아래 항목의 내용만 서술**
* **(기타 내용은 서술하지 않아도 됨. 코드 복사 붙여 넣기 금지)**
* **Task1 (Event-driven Approach with select())**
  + Multi-client 요청에 따른 I/O Multiplexing 설명

Server는 하나의 listenfd와 배열에 저장된 connfd들을 관리한다. 이후 pending input이 있는 fd들을 select 함수로 확인한 후 listenfd에 들어온 요청은 add\_client 함수를 통해 connfd 배열에 새로운 fd를 추가하는 것으로 해결하고, connfd 배열 중 하나의 fd에 들어온 요청(show, buy, sell)은 check\_clients 함수를 통해 그에 맞는 작업을 수행하는 것으로 해결한다. 이 때 select는 인자로 받은 pool.maxfd+1 만큼의 fd들을 관찰하여 I/O Multiplexing을 구현하는 역할을 한다.

* + epoll과의 차이점 서술

select의 경우 대상이 되는 fd들을 모두 확인해야하는 overhead가 발생한다. Epoll 함수는 관찰의 대상이 되는 fd들을 관리하는 저장소를 OS 레벨에서 관리하며, 해당 저장소에 관한 fd를 요청하여 전체 fd를 순회하는 문제점을 해결할 수 있다.

* **Task2 (Thread-based Approach with pthread)**
  + Master Thread의 Connection 관리
  + Worker Thread Pool 관리하는 부분에 대해 서술
* **Task3 (Performance Evaluation)**
  + 얻고자 하는 metric 정의, 그렇게 정한 이유, 측정 방법 서술
  + Configuration 변화에 따른 예상 결과 서술
  1. **개발 방법**
* **B.의 개발 내용을 구현하기 위해 어느 소스코드에 어떤 요소를 추가 또는 수정할 것인지 설명. (함수, 구조체 등의 구현이나 수정을 서술)**

기본적인 구현의 틀은 강의자료를 참조하였다. (Event-driven : ch6 p.19~20, Thread-driven : ch8 p.7~18)

* 공통

주식 정보를 저장하는 구조체인 item을 선언하였다. 주식의 정보(ID, 수량, 가격) 외에 변수의 sharing에서 발생할 수 있는 문제를 방지하기 위한 readcnt와 semaphore인 mutex, w를 가지고 있다.

주식 정보를 저장하는 tree는 binary search tree로 구성하였다. Node 구조체에는 item 정보를 담은 data와 left, right pointer를 가진다.

Tree에 관련된 함수로는 새로운 노드를 삽입하는 insertNode, buy나 sell의 대상을 찾는 search를 구현하였다.

Buy, sell 함수는 입력받은 주식의 ID와 수량을 바탕으로 거래를 진행하며, 그 결과를 요청한 client의 connfd에 전달하는 방식으로 구현하였다.

Show 함수는 tree를 inorder 방식으로 순회하며 출력을 위한 showBuf에 각 node의 정보를 입력해둔 뒤 한번에 connfd에 전달하는 방식으로 구현하였다.

Stock.txt의 정보를 갱신하는 updateFile 함수는 클라이언트의 연결이 종료될 경우 호출되며, 마찬가지로 inorder 방식으로 순회하며 node별로 정보를 저장한다. 이 과정에서 ID를 기준으로 오름차순으로 정렬된다.

처음 stockserver를 실행하면 stock.txt의 정보를 읽어와서 줄 별로 item 구조체 변수에 정보를 담아 insertNode로 저장한다. 이후 각 방식에 맞게 client의 요청을 처리한다.

* Task 1: Event-driven Approach

1. **구현 결과**

* **2번의 구현 결과를 간략하게 작성**
* **미처 구현하지 못한 부분에 대해선 디자인에 대한 내용도 추가**

1. **성능 평가 결과 (Task 3)**

* **강의자료 슬라이드의 내용 참고하여 작성 (측정 시점, 출력 결과 값 캡처 포함)**