[SSE2033: System Software Experiment II - Fall 2020]

Programming Assignment #3

Due: 17th Dec. (Thursday), 23:59:59

1. Introduction

리눅스 환경에서 동작하는 ticketing server 를 제작한다. 이 과제의 주요 목표는 수업에서 배운 주제인 file, process, socket, thread, lock 을 다루는 프로그램을 제작하는 것이다.

2. Problem specification

다수의 client 를 동시에 처리할 수 있는 ticketing server 를 구현한다. Server 는 각 client 의 ticketing 과정과 좌석 위치를 관리한다. Client 는 원하는 좌석에 대한 ticket 을 얻기 위해 server 에 query 를 보낸다. Single query 는 client 가 수행하고자 하는 action 을 포함한다. Server 와 client 및 query 에 대한 정의는 다음과 같다.

2.1. Server

Server 는 다음과 같은 조건에서 infinite loop 를 수행하며 client 로부터 query 를 수신한다.

- Server 로 관리하는 총 좌석의 수는 256 석이다.
- Server 의 각 좌석은 최소한 하나의 synchronization mechanism 으로 관리되어야 한다.
- Server 에서 하나의 synchronization variable 은 하나의 좌석에만 사용되어야 한다.
- Server 는 최대 1024 개의 client 를 동시에 처리할 수 있어야 한다.
- Server 와 client 가 accept 를 통해 성공적으로 연결되면 해당 client 를 위한 전용 thread 를 생성해야 한다. 생성된 thread 는 해당 client 에 대한 모든 query 를 처리해야 한다. 해당 동작을 수행하는 thread 를 미리 생성해 두는 것은 금지된다.
- Server 는 각 "user"의 로그인 상태를 관리한다. "user"는 "client"와는 다르다. "client"는 server 와 통신하는 데 사용되는 program 혹은 process 를 의미하며, 해당 client program 은 특정 "user"의 query 를 server 로 보낼 수 있다.
- User 는 single integer (user ID)로 식별한다.
- User 1 이 현재 client A 를 통해 로그인 되어 있는 경우, 다른 client 에서 user 1 로 로그인 하려는 모든 시도는 server 에 의해 차단되어야 한다. 또한 client A 는 user 1 이 로그아웃 하기 전에 다른 user 의 query 를 server 로 보낼 수 없다. User 1 이 로그아웃 한 이후에, client A 는 다른 user 의 query 를 처리할 수 있다.
- User 가 좌석 예약을 시도할 때 server 는 해당 user 가 로그인 되어 있는지 확인해야 한다.
- Server 는 query 를 수신하고 해당 query 에 대한 결과를 response code 로 client 에 전송한다. 각 query 에 대한 처리 세부사항은 2.3 에서 설명한다.
- Server termination 에 대한 조건은 2.4 에서 설명한다.

2.2. Client

Client 에서 server 로 보내는 single query 는 다음과 같은 structure 로 구성된다:

```
struct query {
       int user;
       int action;
       int data;
};
```

- user field 는 user 의 ID 를 의미하며, 0 ~ 1023 의 범위를 갖는다.
- action field 는 action ID 를 의미하며, 1 ~ 5 의 범위를 갖는다. (0 은 종료 시 사용된다.)
- data field 는 각 action 시에 추가 정보가 필요한 경우 해당 정보를 포함한다.
- 만약 user, action, data field 의 값이 지정된 범위를 벗어나는 경우, server 는 -1 을 return 한다.
- Client 는 다음의 2 가지 방법으로 실행될 수 있다.
    - 실행 시, query file 을 인자로 받는다.
        - Query file 은 1 개 이상의 query 를 갖는다.
        - Query file 을 인자로 받은 client 는 file 내부의 모든 query 를 처리하면 자동으로 termination condition 에 해당하는 query 를 server 로 보내어 종료된다.
    - 실행 시, 인자를 받지 않는다.
        - Client 는 터미널에서 termination condition 에 해당하는 query 가 입력될 때까지 infinite loop 를 수행한다.

```
/* Query File Example */
$ cat query.txt
5, 2, 12
3, 5, 18
```

```
/* Client Example with Query File, check with 2.3.7 */
$ ./client ip_addr port query.txt
Reservation failed
Logout failed
```

```
/* Client Example without Query File, check with 2.3.7 */
$ ./client ip_addr port
5, 2, 12
Reservation failed
3, 5, 18
Logout failed
```

2.3. Action in query

Client 가 single query 를 통해 수행할 수 있는 action 은 다음의 5 가지 종류가 존재한다.

Action ID	Name	Description	data field
1	Log in	로그인 시도	패스워드
2	Reserve	좌석 예약 시도	좌석 번호
3	Check reservation	사용자가 예약한 좌석(들)의 번호를 확인	N/A
4	Cancel reservation	사용자의 예약 취소	좌석 번호
5	Log out	로그아웃	N/A

2.3.1. Log in

- User 가 처음 log in 을 시도하는 경우, 해당 action 은 "registration"으로 처리한다. 이 경우, server 는 query 의 data field 값을 user 의 패스워드로 설정한다. 패스워드는 4 바이트 integer 이다.
- 이미 등록된 user 가 잘못된 패스워드로 log in 을 시도하는 경우, 해당 action 은 fail 처리된다.
- Log in 을 시도하는 user 가 현재 다른 client 로 log in 되어 있는 경우, 해당 action 은 fail 처리된다.
- 위의 두 경우를 제외한 나머지 경우, 해당 action 은 success 처리된다.
- Server 는 log in success 시 1 을 return 하고, fail 시 -1 을 return 한다.

2.3.2. Reserve

- Server 는 user 의 reservation 요청을 처리한다.
- Data field 의 좌석 번호는 0 부터 255 까지이다. 좌석 번호가 범위를 벗어나면, 해당 action 은 fail 처리된다.
- User 가 log in 전에 reserve action 을 시도하면, 해당 action 은 fail 처리된다.
- User 가 reserve 요청한 좌석이 이미 다른 user 에 의해 reserve 된 경우, 해당 action 은 fail 처리된다.
- 위의 세 경우를 제외한 나머지 경우, 해당 action 은 success 처리된다.
- Server 는 reserve success 시 reserve 된 좌석 번호를 return 하고, fail 시 -1 을 return 한다.

2.3.3. Check reservation

- User 가 log in 전에 check reservation action 을 시도하면, 해당 action 은 fail 처리된다.
- User 가 어떠한 좌석도 reserve 하지 않은 경우, 해당 action 은 fail 처리된다.
- 위의 두 경우를 제외한 나머지 경우, 해당 action 은 success 처리된다.
- Server 는 check reservation success 시 전체 좌석 배열을 return 하고, fail 시 -1 을 return 한다.
- 배열의 요소 값은 user 가 reserve 한 좌석에 대해서 1, 그렇지 않은 좌석에 대해서 0 이 입력되어 return 되어야 한다 (배열의 자료형은 자유).
- Client 는 return 된 배열을 읽고, reserve 된 좌석의 번호를 출력해야 한다.

2.3.4. Cancel reservation

- Data field 의 좌석 번호는 0 부터 255 까지이다. 좌석 번호가 범위를 벗어나면, 해당 action 은 fail 처리된다.
- User 가 log in 전에 cancel reservation action 을 시도하면, 해당 action 은 fail 처리된다.
- User 가 어떠한 좌석도 reserve 하지 않은 경우, 해당 action 은 fail 처리된다.
- 위의 두 경우를 제외한 나머지 경우, 해당 action 은 success 처리된다.
- Server 는 cancel reservation success 시 cancel 된 좌석 번호를 return 하고, fail 시 -1 을 return 한다.

2.3.5. Log out

- User 가 log in 전에 log out action 을 시도하면, 해당 action 은 fail 처리된다.
- 위의 경우를 제외한 나머지 경우, 해당 action 은 success 처리된다.
- Server 는 log out success 시 1 을 return 하고, fail 시 -1 을 return 한다.

2.3.6. Response code summary

전체 action 과 그에 해당하는 response 코드는 아래의 표와 같다.

Action ID	Name	On success	On fail
1	Log in	1	-1
2	Reserve	예약된 좌석 번호: [0-255]	-1
3	Check reservation	예약된 모든 좌석 번호: [0, 255]	-1
4	Cancel reservation	예약된 좌석 번호: [0-255]	-1
5	Log out	1	-1

2.3.7. Print format

전체 action 과 그에 해당하는 response 코드에 따른 터미널 출력 내용은 다음과 같다.

Action ID	Name	On success	On fail
1	Log in	Login success	Login failed
2	Reserve	Seat [0, 255] is reserved	Reservation failed
3	Check reservation	Reservation: [0, 255], ...	Reservation check failed
4	Cancel reservation	Seat [0, 255] is canceled	Cancel failed
5	Log out	Logout success	Logout failed

2.4. Termination condition

- Termination query 는 모든 action 을 수행한 client 가 server 로 보낸다.
- Server 가 모든 field(user, action, data)가 0 인 query 를 수신하는 경우, server 는 client 에게 integer type 의 값 256 을 return 한다.
- Server 는 client 에게 256 값을 전송한 이후, 해당 client 와의 연결을 종료한다.
- Client 는 server 로 부터 256 을 return 받은 이후에 실행을 종료한다.

3. Examples of client/server

아래의 두 예시는 client query 예시와 그에 따른 server response 를 보여준다.

	< Client > [user, action, data]	< Terminal >
These queries are not logged in user's query. So not accepted.	Input - 5, 2, 12	Reservation failed
	Input - 3, 5, 18	Logout failed
	Input - 1, 3, 2	Reservation check failed
User 7 login success with password '1234'	Input - 7, 1, 1234	Login success
Not logged in user's query. So failed	Input - 1, 3, 2	Reservation check failed
User 7 reserved seat 12	Input - 7, 2, 12	Seat 12 is reserved
Logged out	Input - 7, 5, 0	Logout success
User 3 login success with password '4321'	Input - 3, 1, 4321	Login success
User 7 already reserved seat 12. So failed	Input - 3, 2, 12	Reservation failed
User 3 reserved seat 21	Input - 3, 2, 21	Seat 21 is reserved
User 3 reserved seat 22	Input - 3, 2, 22	Seat 22 is reserved
User 3 check reservation	Input - 3, 3, 0	Reservation: 22, 23
Logged out	Input - 3, 5, 21	Logout success
Invalid user's password. So failed	Input - 7, 1, 4321	Login failed
Invalid query	Input - 1, 2	Invalied query

< Client 1 >	< Client 2 >	
5, 1, 1		
	5, 1, 1	Failed because user 5 already logged in at Client 1
	3, 1, 7	
5, 2, 10		
	3, 2, 10	Failed because user 5 already reserved seat 10
5, 4, 10		
	3, 2, 10	
5, 5, 1		
	3, 5, 1	
	5, 1, 11	Failed because user 5's password is 1

4. Restrictions

- 과제 구현을 위해 지금까지 배운 리눅스 시스템 콜 / 라이브러리 함수를 이용한다.
- 어떤 자원을 동적으로 할당 받았다면, 프로그램 종료 전에 반드시 해제해야 한다.
  - 여기서 자원이란 file, memory, process, thread, lock 을 뜻한다.
- 본 과제는 개인 과제이다.
- Makefile 과 소스코드 및 프로그램 코드 보고서를 "Student_ID.tar.gz" 형태로 압축하여 icampus 에 제출한다.
  - **컴파일 시, 생성되는 server 프로그램의 이름은 "pa3"로 지정한다.**

    $ tar xvf Student_ID.tar.gz

    $ make

    $ ls | grep pa3

    pa3
  - **제출 서식을 따르지 않거나 제출한 코드가 컴파일 되지 않을 경우, 해당 제출은 채점되지 않는다.**
  - 프로그램 코드와 별도로, 디자인 및 구현에 대한 내용을 담은 보고서를 함께 제출한다. 보고서의 파일 포맷은 pdf 로써, 이름은 student_ID.pdf 로 지정한다.
  - 채점 시간을 기준으로, 가장 최근에 제출된 소스코드에 대해서 채점이 진행된다.
- 과제 제출 시간은 icampus 제출 시각을 기준으로 하며, 과제의 총점은 100 점이며 **매 24 시간마다 10 점**씩 감점된다.