유닉스 프로그래밍

Project 02

컴퓨터과학과

201710924 박도원

주제 설명

주제

쇼핑몰 프로세스

설명

일종의 쇼핑몰 프로세스를 구현하였다. 클라이언트는 상품을 5개씩 조회할 수 있다. 마음에 드는 물건이 있으면 상품 구매를 할 수 있다. 상품 구매 요청 시 서버에서는 클라이언트가 요청한 상품의 개수만큼 있는지 확인하고 구매 가능 여부를 따진다. 가능하다면 주문을, 아니라면 클라이언트에게 주문 재요청을 보낸다. 판매자는 상품 등록, 수정, 삭제를 할 수 있고 누적 판매량 기준으로 상위 3개의 상품을 볼 수 있다. 만약 동시에 구매하려는 사람들이 있다면 mutex를 사용하여 데이터의 무결성을 유지시킨다.

시스템 설계도

함수 - Server

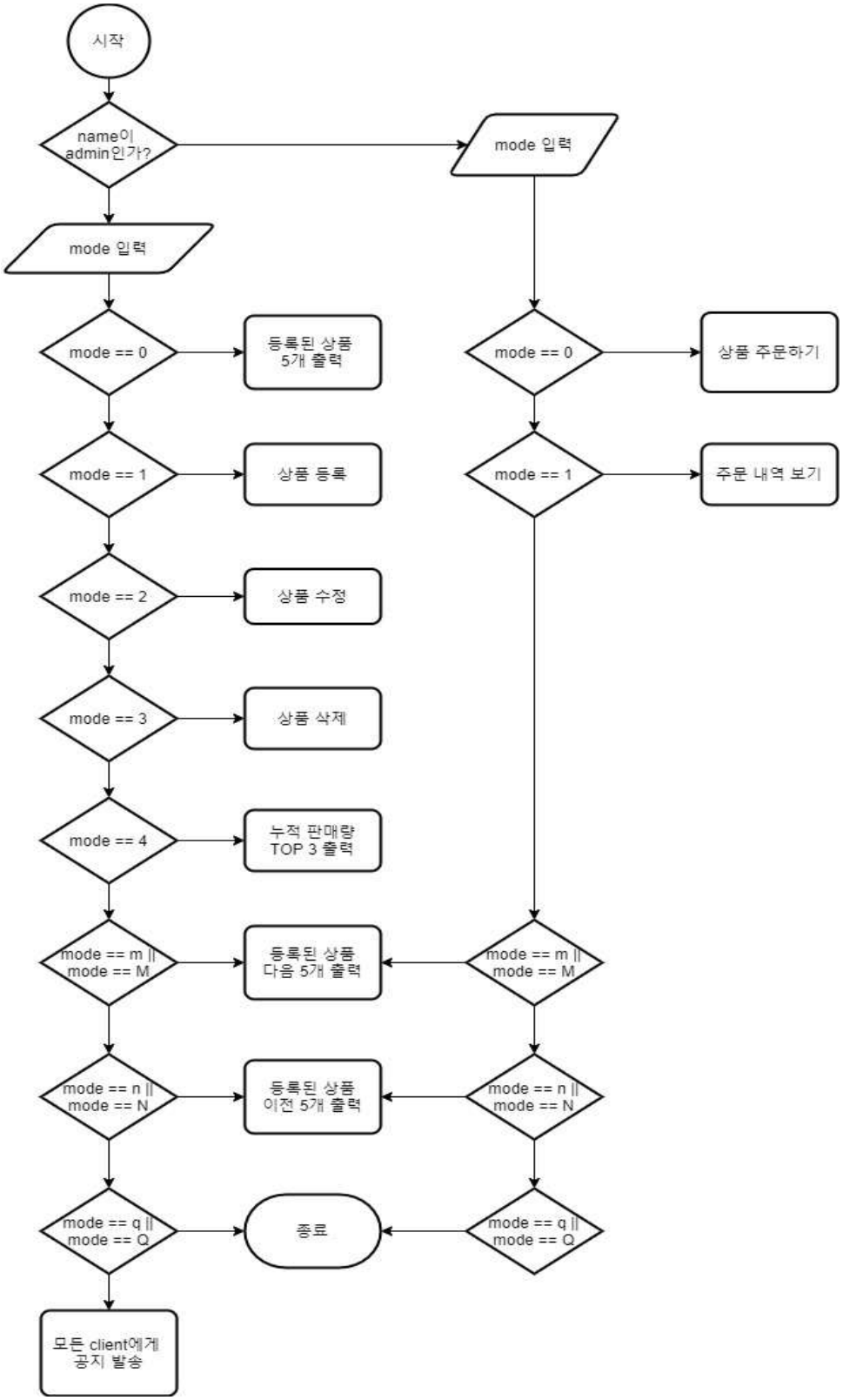
항목	설명
void info()	서버 소켓 생성 시 서버 정보 출력
void *handle_cli(void *arg)	클라이언트 핸들러
void send_msg(int cli_fd, char *msg, int len)	클라이언트에게 보낼 데이터 관련 함수
int process_item(int cli_fd, char *msg)	상품 주문 시 데이터 처리
void print_connect_log(struct sockaddr_in cli_addr)	클라이언트 서버 접속 시 로그 출력
void print_disconnect_log()	클라이언트 접속 해제 시 로그 출력
void process_admin (int cli_fd, char *msg)	admin 기능 데이터베이스 처리 관련 함수
void process_client (int cli_fd, char *msg)	사용자 기능 데이터베이스 처리 관련 함수

함수 - Client

항목	설명
void menu()	쇼핑몰 메뉴 출력
void menu_admin()	쇼핑몰 관리자 메뉴 출력

void process(int sock)	일반 사용자의 입력에 따른 기능 수행
void process_admin(int sock)	관리자의 입력에 따른 기능 수행
void* send_data(void *arg)	서버에 데이터 보냄
void* recv_data(void *arg)	서버에서 데이터 받음
char* insert_(int cli_fd)	관리자 상품 등록 기능
char modify_(int cli_fd)	관리자 상품 수정 기능
char* delete _(int cli_fd)	관리자 상품 삭제 기능
char* view_top3(int cli_fd)	관리자 누적판매량 보기
char* myorder_ (int cli_fd)	사용자 주문 내역 보기
char* view (int cli_fd, int start)	상품 5개씩 보기
char* order(int cli_fd, char* name)	사용자 상품 주문하기
void view_top_3_structure()	누적 판매량 테이블 구조 출력
void view_structure()	상품 테이블 구조와 주문내역 테이블 구조 출력

순서도


시작
name이 admin인가?
mode 입력
mode 입력
mode == 0
등록된 상품 5개 출력
mode == 1
상품 등록
mode == 2
상품 수정
mode == 3
상품 삭제
mode == 4
누적 판매량 TOP 3 출력
mode == m || mode == M
등록된 상품 다음 5개 출력
mode == n || mode == N
등록된 상품 이전 5개 출력
mode == q || mode == Q
종료
모든 client에게 공지 발송
mode == 0
상품 주문하기
mode == 1
주문 내역 보기
mode == m || mode == M
mode == n || mode == N
mode == q || mode == Q

사용된 기술

개발 환경

- WSL (Windows Subsystem for Linux) 2

- Ubuntu 18.04.5 LTS

- gcc 7.5.0

사용된 Library interface

(string.h의 기본 함수들은 명시하지 않음)

- libsqlite3-dev : 상품 데이터 및 주문 데이터 저장 시 사용을 위한 데이터베이스 라이브러리

- socket programming

- server

헤더파일	함수명	설명
pthread.h	pthread_mutex_init(3)	상호배제를 위해 사용할 mutex 초기화
	pthread_mutex_lock (3)	mutex lock
	pthread_mutex_unlock(3)	mutex unlock
sqlite3.h	sqlite3_open (3)	데이터 저장을 위한 sqlite3 database 초기화
	sqlite3_close_v2 (3)	데이터베이스 close
	sqlite3_prepare_v2 (3)	database와 sql문 준비 상태 확인
	sqlite3_step (3)	sqlite3 실행
	sqlite3_bind_text (3)	인자에 text 형태로 넣어줌
	sqlite3_bind_int (3)	인자에 int 형태로 넣어줌
	sqlite3_column_int (3)	결과를 int형태로 받음
	sqlite3_column_text (3)	결과를 text 형태로 받음
	sqlite3_reset (3)	stmt 구조체 초기화
	sqlite3_finalize (3)	stmt 구조체 메모리 할당 해제
string.h	memset(3)	파일 기술자 메모리 설정
sys/types.h sys/socket.h	socket(2)	소켓 파일 기술자 생성
	bind(2)	소켓 파일기술자를 지정된 ip주소, 포트번호와 결합
	listen(2)	클라이언트의 접속 요청 대기
	accept(2)	클라이언트의 접속 허용
	recv(2)	데이터 수신

	send(2)	데이터 송신
unistd.h	close(2)	소켓 파일기술자 종료
pthread.h	pthread_create(3)	tthread 생성
	pthread_detach(3)	thread와 프로세스 분리

- client

헤더파일	함수	설명
string.h	memset(3)	파일 기술자 메모리 설정
sys/types.h sys/socket.h	socket(2)	소켓 파일 기술자 생성
	bind(2)	소켓 파일기술자를 지정된 ip주소, 포트번호와 결합
	connect(2)	클라이언트의 접속 요청
	recv(2)	데이터 수신
	send(2)	데이터 송신
	close(2)	소켓 파일기술자 종료
pthread.h	pthread_create(3)	tthread 생성
	pthread_join(3)	thread에 대한 종료 기다림

사용 메뉴얼

compile

1. sudo apt install libsqlite3-dev : sqlite3 설치

2. server.c 와 client.c 가 있는 폴더 내에 project2.db 파일 만들기 (파일 존재하면 생략)

2-1. sqlite3 project2.db 명령 실행

2-2. sqlite3 내에서 테이블 만들기

- CREATE TABLE item (id integer primary key autoincrement, name text, price int, stock_quantity int, sales int);

- CREATE TABLE orders (id integer primary key autoincrement, name text, itemname text, count integer, total_price integer);

3. gcc -o server server.c -lpthread -lsqlite3 : 서버 컴파일

4. gcc -o client client.c -lpthread : 클라이언트 컴파일

Server

입력	설명
./server	서버 실행

Client

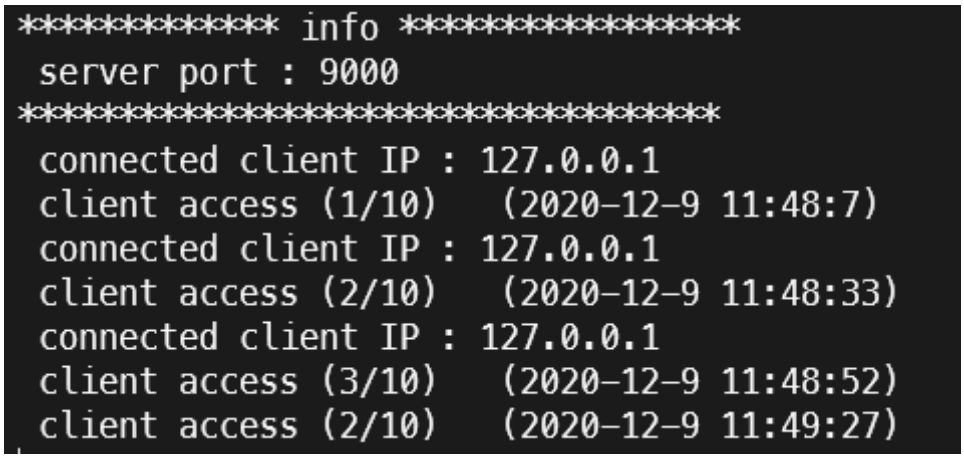
모드	입력	설명
공통	./client IP주소 서버포트번호 사용자이름	사용할 IP주소, 서버 포트번호, 사용자이름을 입력한다. 이름이 admin이면 관리자모드, 아니면 사용자 모드로 들어간다.
공통	m 또는 M	상품 리스트 출력 (다음 5개)
공통	n 또는 N	상품 리스트 출력 (이전 5개)
공통	q 또는 Q	클라이언트 종료
사용자	0	상품 주문 모드
사용자	1	주문 내역 출력 모드
관리자	0	상품 목록 출력
관리자	1	상품 등록 모드
관리자	2	상품 수정 모드
관리자	3	상품 삭제 모드
관리자	4	상품 누적 판매량 TOP3 출력 모드

사용 예시

* Server

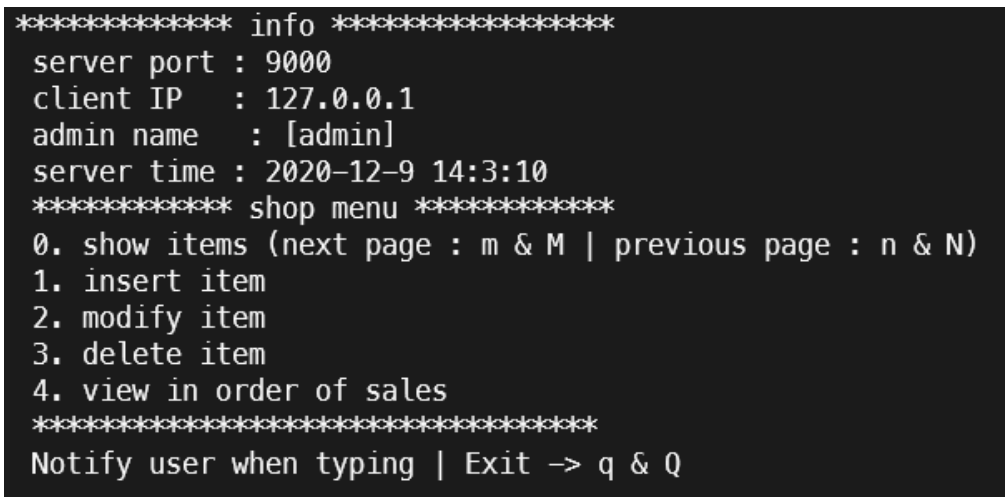
서버 실행

클라이언트 접속 및 연결 해제 시 서버 기록



* Client - admin

클라이언트 실행



공지 보내기

```
test
[notice] test
```

상품 등록 성공 (name 입력 시 공백 문자는 넣으면 안됨)

```
1
*****insert item*****
item name : insert_test
item price : 2000
item stockQuantity : 10
success about database item.
```

상품 수정 성공

```
2
*****modify item*****
item name : insert_test
modify item price : 100
modify item stockQuantity : 500
success about database item.
```

상품 수정 성공 후 상품 목록

```
********ITEMS - 2********
------------------------
| name  | amount | price |
------------------------
| test2 | 48     | 50    |
| nae   | 30     | 90    |
| pin2  | 21     | 300   |
| dowon | 2      | 4     |
| insert_test   | 500    | 100   |
```

상품 삭제 성공

- 삭제 전 상품 리스트

```
********ITEMS - 2********
------------------------
| name  | amount | price |
------------------------
| test2 | 48     | 50
| nae   | 30     | 90
| pin2  | 21     | 300
| dowon | 2      | 4
| insert_test   | 500    | 100
```

- 삭제

```
3
*****delete item*****
delete item name : insert_test
success about database item.
```

- 삭제 후 상품 리스트

```
m
********ITEMS - 2********
------------------------
| name   | amount | price |
------------------------
| test2 | 48     | 50
| nae    | 30     | 90
| pin2   | 21     | 300
| dowon | 2      | 4
```

누적 판매량 TOP3 확인

```
4
***view in order of sales***
***TOP 3 ITEMS***
------------------------------
|name    | sales | total_price |
------------------------------
| Python | 6 | 60000
| Go | 4 | 96000
| java | 2 | 40000
------------------------------
```

* Client - member

클라이언트 실행

```
********** info **********
 server port : 9000
 client IP   : 127.0.0.1
 name        : [dw]
 server time : 2020-12-9 15:38:22
 ********** shop menu **********
 0. want to order
 1. show my orders
 ********************************
 next page : m & M | previous page : n & N | Exit -> q & Q

*******ITEMS - 1*******
------------------------
| name  | amount | price |
------------------------
| java | 3 | 20000
| C++ | 10 | 15000
| Python | 20 | 10000
| Swift | 3 | 25000
| Go | 15 | 24000
```

다음 페이지 상품들 보기

```
m
*******ITEMS - 2*******
-------------------------
| name  | amount | price |
-------------------------
| unix_programming | 2 | 30000
```

이전 페이지 상품들 보기

```
n
*******ITEMS - 1*******
-------------------------
| name  | amount | price |
-------------------------
| java | 3 | 20000
| C++ | 10 | 15000
| Python | 20 | 10000
| Swift | 3 | 25000
| Go | 15 | 24000
```

주문한 상품 보기

```
1
******MY ORDER ITEMS******
------------------------
| name  | amount | price |
------------------------
| java | 2 | 40000
```

상품 주문하기

```
0
item name : Go
quantity : 4
success order : Go - 4 | total : 96000
```

상품 주문 후 결과

```
1
******MY ORDER ITEMS******
------------------------
| name  | amount | price |
------------------------
| java | 2 | 40000
| Go | 4 | 96000
```

없는 상품 주문 시 결과

```
0
item name : s
quantity : 2
해당 상품이 존재하지 않습니다.
```

주문량이 재고량보다 많은 경우

```
0
item name : Python
quantity : 25
구매할 수 있는 수량 : 20
```

admin에서 공지 보낸 후 결과 - admin을 포함한 모든 client에게 전송

```
[notice] test
```