리눅스 프로그래밍 보고서

키오스크 시스템

제출일	2023. 06. 14.	전 공	컴퓨터공학부
과 목	리눅스 프로그래밍	조	7조
담당교수	이세호 교수님	팀 원	김채현, 이은정, 홍사강

목 차

I. 서 론

1. 프로젝트 주제 및 목적

1) 프로젝트 주제

(1) 본 보고서는 상점 키오스크 시스템을 구현하기 위한 리눅스 기반 서버-클라이언트 프로그램에 대한 설명이다. 이 프로젝트는 서버와 클라이언트 간의 상호작용을 통해 키오스크의 관리와 사용을 가능하게 하는 시스템을 구현하였다. 이를 통해 고객은 키오스크를 통해 상품을 선택하고 구매할 수 있으며, 서버는 상품 관리와 재고 관리 등의 기능을 수행한다. 이 보고서에서는 프로젝트의 개요, 구현된 기능, 프로그램의 특징 및 장점 등을 설명할 것이다.

2) 프로젝트 목적

(1) 본 프로젝트의 목적은 우리 7조 조원들이 리눅스 시스템 호출을 활용하여 실제로 서버-클라이언트 모델을 구현하고, 키오스크 시스템의 동작을 이해하는 것이다. 이 프로젝트를 통해 학생들은 시스템 호출을 활용한 프로그래밍과 네트워크 통신 개념에 대한 이해를 도모하고, 실제 상업적 환경에서 사용되는 키오스크의 동작 원리를 학습할 수 있다. 또한, 다중 접속, 재고 관리 등의 추가 기능 구현을 통해 조원들의 리눅스 프로그래밍 역량을 향상시키고, 문제 해결과 협업 능력을 향상시켜 조원 뿐만 아니라 리눅스 수강자 학생들 앞에서 발표하며 학생들의 이해의 지평을 넓히는 데에 의의가 있다.

2. 제작 목표

1) 키오스크의 기본 기능은 물론, 팀 내 많은 아이디어 공유를 통하여 특별한 추가적인 기능을 구현한다.

(1) 장바구니 기능: 장바구니를 이용하여 물건을 넣고 빼는 등의 동작을 수행할 수 있도록 한다.

(2) 개별 장바구니 관리: 각 클라이언트마다 자신만의 장바구니를 생성하고 관리할 수 있도록 한다.

(3) 다중 카테고리 지원: 상품을 카테고리별로 나누어 관리할 수 있도록 한다.

(4) 다중 클라이언트 접속: 여러 클라이언트가 동시에 서버에 접속 할 수 있도록 한다.

(5) 페이지 구현: 여러 페이지를 나누어 한 페이지에 아이템이 10개씩만 보이도록 함으로써 키오스크 내용이 한눈에 들어오도록 효과적으로 표시할 수 있도록 한다.

(6) 실시간 재고 반영: 장바구니에 물건을 담을 때 재고가 실시간으로 반영되어 다중 접속 클라이언트 사이에서 재고 상황이 공유되도록 한다.

(7) 모듈화를 위한 manage.h와 manage.c 파일: 조원들이 개별적으로 개발한 함수들을 헤더 파일을 사용하여 하나의 프로그램으로 원활히 합칠 수 있도록 한다.

(8) 파일 처리 개선: 파일과 관련된 처리에서 모두 쓴 파일을 닫는 작업을 확실히 하기 위해, 함수를 사용하여 열고 닫는 작업을 일관되게 처리한다.

3. 프로젝트 진행 일정 (제작 기간)

(1) 05/22: 프로그램 구조 구상, 키오스크 프로그램을 구현하기 위해 필요한 파일들 구상(item_update.c 등), 그 파일들을 어떻게 상호작용화 할 것인가에 대한 구상, 어떠한 추가 기능을 구현할 수 있을까에 대한 브레인 스토밍, 출력 화면 구상

(2) 05/27: 서버 파일, 클라이언트 파일 등 파일에 들어갈 함수들(상품 진열 함수, 장바구니 계산 함수, 시작화면 아스키 아트 출력, 소켓 생성 및 종료 등)에 대한 구상, 데이터 파일 구조체 구상(수량, 가격, 이름, 카테고리)

(3) 05/29: 아이템, 카테고리 구조체 생성, 장바구니와 상품관련 파일 구현, 클라이언트 파일 일부분 구현, 소켓 통신 다중 접속 서버 공부, 서버와 소켓 통신 등 주고받는 데이터 형식(이진파일)과 클라이언트 파일에서의 활용(장바구니 출력 방식, 클라이언트 파일에서의 데이터 입력 등)을 고려해서 데이터 파일 구현

(4) 06/03: 시작화면창, 상품진열창, 장바구니 확인창, 상품세부사항창, 결제창 디자인 및 구현

(5) 06/06~06/10: 상호작용의 주축인 서버, 클라이언트, manage 파일 구현 및 오류 수정
(6) 06/11: 프로그램 완성 및 제출
(7) 06/14: 프로젝트 발표

II. 본 론

1. 프로젝트 개요 및 흐름

1) 프로젝트 개요

(1) 키오스크 프로그램은 리눅스 기반의 서버-클라이언트 모델로 구현된 시스템이다. 이것은 서버와 클라이언트 간의 상호작용을 통해 키오스크의 관리와 사용을 가능하게 한다. 서버는 키오스크를 관리하는 주체로서, 상품 정보(상품 종류, 가격, 수량)를 제공하고 각 상품의 수량을 관리한다. 또한, 수량 초과 시 클라이언트가 상품을 가져갈 수 없도록 한다. 상품 총 금액과 지불 금액을 비교하여 각 사용자에 대한 계산을 수행한다. 클라이언트는 키오스크의 고객 역할을 수행한다. 원하는 상품에 접근하고 상품의 개수를 입력한 뒤 필요한 상품을 선택하여 장바구니에 담는다. 클라이언트는 페이지 당 10개씩의 아이템 목록을 볼 수 있으며, 장바구니에 물건을 담을 때 해당 상품의 재고 상황이 실시간으로 반영된다. 또한, 클라이언트마다 개별적인 장바구니를 할당받아 물건을 담고 뺄 수 있다. 프로그램은 다중 접속을 지원하여 여러 클라이언트가 동시에 사용할 수 있도록 한다. 또한, manage.h와 manage.c 파일을 통해 조원들이 나눠서 개발한 함수들을 효율적으로 관리하고 합칠 수 있도록 한다. 클라이언트마다 개별 장바구니 할당, 카테고리별 상품 분류, 다중 접속 지원 등 추가 기능들을 제공하도록 한다.

2) 프로젝트 흐름

(1) 서버 시작 및 초기화

: server.c 파일에서 main() 함수가 시작된다. 소켓 생성 및 주소 설정이 이루어지고, 서버 소켓은 클라이언트의 연결을 대기한다.

(2) 클라이언트 연결 처리

: accept() 함수를 통해 클라이언트의 연결을 수락한다. fork()를 통해 클라이언트와의 통신을 처리하는 자식 프로세스를 생성한다.

(3) 클라이언트 요청 처리(후에 자세히 서술)

: 클라이언트와의 통신은 read()와 write() 함수를 사용하여 이루어진다. 클라이언트는 페이지 식별자를 보내어 서버에 요청을 전달한다. 서버는 받은 페이지 식별자에 따라 적절한 동작을 수행한다.

(4) 시작화면 (A 페이지)

: 서버는 A 페이지를 클라이언트에 전송하여 시작 화면을 출력한다.

(5) 카테고리 페이지 (C 페이지)

: 클라이언트는 원하는 카테고리를 선택하여 서버에 전달한다. 서버는 해당 카테고리의 상품 목록을 클라이언트에 전송한다. 클라이언트는 상품을 선택하고 수량을 입력하여 장바구니에 추가한다.

(6) 상품 목록 (P 페이지)

: C 페이지에서 받아 저장한 box 파일(현재 카테고리의 재고)로부터 읽어서 상품 목록을 출력한다.

(7) 상품 상세 (D 페이지)

: box 파일로부터 원하는 상품의 상세 정보를 요청한다.

(8) 장바구니 (B 페이지)

: 클라이언트는 장바구니 페이지에서 상품의 수량을 선택하고 결제를 진행한다. 서버는 클라이언트가 선택한 상품의 수량을 업데이트하고 재고 여부를 확인한다.

(9) 결제 (L 페이지)

: 클라이언트는 결제 페이지에서 상품 총 금액과 지불 금액을 비교하여 결제를 완료한다.

(10) 서버 종료

: 클라이언트가 연결을 종료하면 해당 클라이언트와 관련된 프로세스는 종료된다. 서버는 계속해서 다른 클라이언트의 연결을 수락하고 처리한다.

=> 이러한 흐름을 통해 서버는 키오스크 관리 및 상품 선택, 수량 업데이트, 결제 등의 기능을 수행하며, 클라이언트는 키오스크 고객으로서 상품 탐색, 결제 등을 수행한다.

2. 파일 별 세부 사항

1) client.c 파일: 상품에 접근, 상품 개수 입력, 상품 선택 및 결제.

(1) 기능

* 소켓 생성 및 서버 연결: socket() 함수를 사용하여 소켓을 생성하고, connect() 함수를 사용하여 서버에 연결한다. 이를 통해 클라이언트는 서버와 통신할 수 있다.

* 시작 페이지(A)와 카테고리 목록(C) 표시: 서버에 시작 페이지 또는 카테고리 목록을 요청하고, 그에 따라 화면에 출력한다. 고객은 카테고리를 선택할 수 있다.

* 상품 목록(P) 표시: 선택한 카테고리에 해당하는 상품 목록을 서버에 요청하고, 받아온 정보를 화면에 출력한다. 고객은 아이템을 선택하거나 아이템의 상세 정보를 확인할 수 있다.

* 상품 상세 정보(D) 표시: 선택한 상품의 상세 정보를 서버에 요청하고, 받아온 정보를 화면에 표시한다. 고객은 해당 상품을 장바구니에 추가하거나 이전 화면으로 돌아갈 수 있다.

* 장바구니(B) 표시 및 관리: 장바구니에 담긴 상품 목록을 서버에 요청하고, 받아온 정보를 화면에 표시한다. 고객은 장바구니에서 상품을 삭제하거나 결제를 진행할 수 있다.

* 결제(L): 결제하기 전에 장바구니에 담긴 상품 목록과 총 결제 금액을 화면에 출력한다. 고객은 결제를 위해 금액을 입력하고, 필요한 경우 거스름돈을 받을 수 있다.

* 다중 클라이언트 지원: getpid() 함수를 사용하여 각 클라이언트의 프로세스 ID를 얻고, 해당 ID를 기반으로 장바구니 파일을 개별적으로 생성하여 관리한다.

2) item.h 파일: 상품에 대한 구조체 및 변수 선언.

(1) 기능

* 상품 정보를 관리하기 위한 구조체(struct item)을 정의한다.

* 이 구조체는 아이템의 ID, 이름, 가격, 재고 수량, 정보를 저장하는 변수들을 포함한다.

* 이 헤더 파일은 상품 관리를 위한 기능들을 구현하기 위해 사용된다. 이를 통해 프로그램은 상품 목록을 출력하고, 상품 정보를 업데이트하며, 재고를 관리할 수 있게 된다.

3) item_create.c 파일: 상품 생성 함수 구현. 새로운 상품 추가 시 사용

(1) 기능

1. 상품 정보 입력
    * scanf() 함수를 사용하여 상품의 이름, 가격, 수량, 정보를 입력받는다.
2. 상품 ID 계산 및 파일에 기록
    * 상품의 ID는 파일에 저장된 이전 상품 개수에 1을 더한 값으로 계산된다.
    * lseek() 함수를 사용하여 파일 포인터를 상품 ID에 해당하는 위치로 이동시킨다.

* write() 함수를 사용하여 상품 구조체를 파일에 기록한다.

4) item_query.c 파일: 상품 정보 조회 함수 구현.

(1) 기능

1. 파일 열기 및 오류 처리
    * 입력된 카테고리 파일을 이진 모드로 읽기 위해 fopen() 함수 사용
2. 상품 검색
    * 입력된 ID를 사용하여 fseek() 함수를 통해 해당 상품의 위치로 파일포인터를 이동 시킨다.
    * fread() 함수를 사용하여 파일로부터 상품 구조체를 읽어온다.
    * 읽어온 상품이 존재하고 ID가 0이 아니면, 상품의 정보를 출력한다.
    * 상품이 존재하지 않는 경우, "item %d 없음" 메시지를 출력한다.

5) item_update.c 파일: 상품 수량 관리 함수 구현.

(1) 기능

1. 상품 업데이트
    * 사용자로부터 업데이트할 상품의 ID를 입력받는다.
    * 입력된 ID를 사용하여 fseek() 함수를 통해 해당 상품의 위치로 파일 포인터를 이동시킨다.
    * 상품을 성공적으로 읽어온 경우, 해당 상품의 정보를 출력한다.
    * 사용자는 상품의 수량, 가격, 이름, 정보 중 하나를 선택하여 업데이트 할 수 있다.
    * 파일 포인터를 이동시켜 현재 위치의 상품을 수정하고, fwrite() 함수를 사용하여 업데이트 된 상품 정보를 파일에 쓴다.
2. 새로운 상품 생성
    * 상품이 존재하지 않는 경우, 사용자에게 해당 ID로 새로운 상품을 생성할 것인지 물어본다.
    * Y라고 응답할 경우, create() 함수를 사용하여 빈 상품을 새로 생성하고 해당 ID로 저장한다.

6) manage.c 파일: 모듈화 위해 만든 파일.

(1) 기능

1. create: 상품이나 카테고리를 생성하는 함수. 입력받은 정보를 바탕으로 새로운 레코드를 생성하고 파일에 저장한다.

2. query: 특정 카테고리에서 지정된 ID와 수량에 맞는 상품이 있는지 확인하는 함수. 수량이 충분한 경우 해당 상품의 수량을 반환하고, 부족한 경우 –1을 반환한다.

3. getname: 특정 카테고리에서 지정된 ID에 해당하는 상품의 이름을 가져오는 함수. 해당 상품이 존재하는 경우 이름을 반환하고, 존재하지 않는 경우 NULL을 반환한다.

4. getprice: 특정 카테고리에서 지정된 ID에 해당하는 상품의 가격을 가져오는 함수. 해당 상품이 존재하는 경우 가격을 반환하고, 존재하지 않는 경우 –1을 반환한다.

5. stockUpdate: 특정 카테고리에서 지정된 ID에 해당하는 상품의 재고를 업데이트하는 함수. 파일을 열어 해당 상품의 레코드를 찾고, 새로운 재고 수량으로 업데이트한 후 파일에 저장.

6. showBasket: 장바구니에 담긴 상품 목록을 출력하는 함수. 장바구니 파일을 열어서 각 상품의 정보를 출력하고, 총 수량과 가격을 계산하여 출력한다.

7. name_query: 지정된 카테고리에서 이름을 기준으로 상품을 검색하는 함수. 입력받은 이름과 일치하는 상품의 ID를 반환한다.

8. startFrame: 프로그램 시작 시 화면에 출력되는 시작 프레임을 구성하는 함수. 아스키 아트로 디자인한

키오스크 로코와 안내 문구를 출력한다.

9. printCategory: 카테고리 목록을 출력하는 함수. 아스키 아트와 예쁜 이모티콘으로 디자인한 카테고리 문구와 고객으로부터 원하는 카테고리를 입력받을 수 있다.

10. displayproduct: 특정 카테고리에서 상품 목록을 화면에 출력하는 함수이다. 한 페이지 당 박스에 담긴 아이템이 10개씩 출력되도록 디자인하였다.

7) server.c 파일: 서버 역할 수행. 키오스크 관리와 서비스 제공 담당.

(1) 기능

1. 서버 소켓 생성 및 설정: socket() 함수를 사용하여 AF_UNIX 소켓을 생성하고, bind() 함수를 통해 서버 주소를 할당한다.

2. 연결 대기: listen() 함수를 사용하여 클라이언트의 연결 요청을 대기한다.

3. 클라이언트와의 통신 처리: accept() 함수를 사용하여 클라이언트의 연결을 수락하고, fork()를 통해 클라이언트와의 통신을 처리하는 자식 프로세스를 생성한다. 이후 클라이언트로부터 받은 요청에 따라 적절한 동작을 수행한다.

4. 페이지별 동작 처리: 클라이언트로부터 받은 페이지(상황별 화면) 식별자에 따라 해당 페이지의 동작을 처리한다.

5. 통신 및 데이터 처리: read()와 write() 함수를 사용하여 클라이언트와 데이터를 주고받는다.

6. 장바구니 업데이트: 클라이언트로부터 받은 상품 ID와 수량을 이용하여 장바구니에 상품을 추가하거나 재고를 업데이트 한다. 이를 위해 stockUpdate() 함수를 호출한다.

III. 결 론

1. 주요 추가 기능

1) 클라이언트별로 장바구니를 생성해서 각각을 관리한다.

client.c:

```
int pid = getpid();
 sprintf(basketName, "basket%d", pid);
  printf("%d\n", pid);
 FILE* basket = fopen(basketName, "wb+");
 fclose(basket);
```

2) 카테고리를 쪼개어 각각을 따로 나눌 수 있다.

```
(카테고리 하나당 개별 이진파일로 만듦)
scanf("%s", clientMsg);
 // 카테고리 재고 받아오기
 write(clientfd, clientMsg, strlen(clientMsg)+1); // 카테고리명 넘기기

server.c:
…
printf("카테고리 페이지\n");
write(connfd, &page, sizeof(page));
```

```
readLine(connfd, category);
printf("%s\n", category);
sendBox(connfd, category);
```

3) 클라이언트 다중 접속이 가능하다.

server.c:

```
        while (1) {
 connfd = accept(listenfd, (struct sockaddr*)&clientAddr, &clientLen);
  if (fork() == 0)
```

4) 페이지를 구현하여 상품을 페이지 단위로 볼 수 있다.

manage.c

```
int displayproduct(char* category, int start) {
…
int i = start;
int end = start + 10; // 한 페이지당 10개
fseek(fp, (long) (start-START_ID)*sizeof(record), SEEK_SET);
while ((fread(&record, sizeof(record), 1, fp) > 0) && (i < end)) {
…
if (record.quantity > 0) {…}
else // 재고가 없을 땐 empty
        {
   printf("\t%-12s", "empty");
  i++;
 }
…
}
if (i != end) { end = i; …} // 10개 찍기 전에 끝날 때 end 업데이트
…
return end; // 다음 출력을 위해 끝 번호 반환
```

client.c:

```
…
end = displayproduct(boxName, start);
printf("넘기기(B, N), 아이템 상세(D), 아이템 담기(+), 카테고리(C): ");
…
scanf(" %c", &ans);
if (ans == 'N')
{
    if (end % 10 != 1) // 마지막 페이지라 못 넘김
    else { start = end; index++; } // 다음 페이지로
}
else if (ans == 'B')
{
    if (index == 1) // 첫 페이지라 못 넘김
```

```
else { start -= 10; index--; } 이전 페이지로
}
```

5) 장바구니에 물건을 담을 때, 재고에 반영되어 다중 클라이언트 사이에서 재고 상황이 공유된다.

```
server.c:
…
read(connfd, &id, sizeof(id));
read(connfd, &quantity, sizeof(quantity));
printf("%d %d", id, quantity);
int cid = stockUpdate(category, id, quantity);
if (cid == id)
{ printf("재고 반영 성공\n"); }
```

6) 장바구니를 이용하여 상품을 추가 및 제거할 수 있다.

```
printf("아이템 담기(+), 아이템 목록(P): ");
…
scanf(" %c", &ans);
  if (ans == '+') {… // 재고 존재 여부 체크
if (id < 0 || query(boxName, id, count) == -1) {…}
else { … // 필요한 정보 가지고 장바구니 페이지로 이동 }
  … // 장바구니 페이지에서 재고 반영
```

7) 함수를 모듈화하여 파일 관련 처리를 효율적으로 하였다.

```
manage.c, manage.h
```

8) 파일과 관련된 처리를 위해 함수 내에서 파일의 오픈 및 클로즈를 확실히 처리하였다.

```
manage.c:
/* send box */
int sendBox(int destfd, char* filename)
{
    FILE* fp = fopen(filename, "rb"); // 파일 열기
… // 데이터 전송 수행
printf("close:%d\n", fclose(fp)); // 파일 닫고 결과 출력
 return result;
}
```

2. 제작 후기 및 고찰

: 프로젝트 진행 과정에서 조원들과의 협업과 팀워크를 통해 많은 것을 배웠다. 조원들과의 적극적인 의사 소통과 역할 분담은 프로젝트의 성공을 위한 핵심적인 요소였다. 문제 해결 과정에서 서로의 아이디어를 공유하고 토론하며 최적의 해결책을 찾는 경험은 꽤나 힘들었지만 이를 통해 개인적인 성장과 조원들과의 긍정적인 상호작용 및 유대감을 쌓을 수 있었다. 프로그램 제작 후 제3자의 입장에서 바라보고, 다른 친구들로부터 피드백을 수집하는 과정에서, 고객이 기대하는 기능과 요구사항을 더 정확하게 파악하는 것이 중요함을 깨달았다. 피드백 중 하나로는, 중간에 종료할 수 있거나 꼭 상품을 담는 액션을 취하지 않아도 장바구니를 볼 수 있게 구현하는 등 페이지 흐름이 조금 더 매끄러웠으면 좋았을 것 같다는 의견이 있었다. 이 피드백처럼 사용자 중심의 접근 방식을 적용하여 사용자들의

의견을 적극 수용하고 필요한 기능을 추가하거나 개선해야할 필요성을 느꼈다. 또한, 페이지 파악을 위해 서버와 클라이언트가 page를 주고 받는 이런 반복되는 부분들도 함수로 처리할 수 있었을 텐데 그러지 못해서 코드 가독성이 조금 떨어진다고 판단되어 아쉬웠고, 한 번에 필요한 개별 정보에 대해서는 개별로 주고 받기보다 구조체를 만들어서 한꺼번에 주고 받았으면 더 좋았을 것 같다는 생각을 하고 개선의 여지를 두었다.

3. 역할 분담

1) 김채현: 프로젝트 아이디어 제시, 코드 구현 및 디버깅 오류 수정, 프로젝트 구조 설계에 대한 의견 제시 등의 역할을 수행. 발표 담당.

2) 이은정: 프로젝트 아이디어 제시, UI 디자인 및 코드 구현, 구조 설계에 대한 의견 제시. 보고서 작성.

3) 홍사강: 프로젝트 아이디어 제시, 코드 구현 및 개별 구현된 함수 취합, 디버깅 및 오류 수정, 구조 설계에 대한 의견 제시, 데이터 파일 입력.