課題要旨

本課題では，自販機のサーバプログラムを作成する．複数クライアントからの接続に対して，`select()` を用いて，入出力を多重化する．

処理概要

データ構造は，以下のとおりである．クライアントデータ構造には，ソケットディスクリプタ，投入金額，現在購入しようとしている商品構造体のポインタを持つ．クライアントの最大接続台数は 5 台とし，それぞれのクライアントデータ構造を配列で持つ．商品データ構造体のポインタを持つことで，購入情報の冗長化を防ぐ．

Listing 1　商品構造

```
typedef struct {
    char *name;
    int price;
    int stock;
} Product;
```

Listing 2　クライアント構造

```
typedef struct {
    int clSock;
    int coin;
    Product *drinkProduct;
} Client;
```

サーバは，自身のソケットを作成し，クライアントからの接続を待ち受ける．ここで，`select()` を用いて，複数のクライアントからの接続を待ち受ける．その際，デバッグ用として，標準入力を受け付けるため，`stdin` も監視対象に加える．

`stdin` からの入力があった場合　入力が `ls` ならば，在庫一覧を表示する．また，入力が `cls` なら，接続中のクライアントリストを表示する．EOF が入力された場合には，サーバを終了する．この時，接続中のクライアントにも終了を通知し，ソケットを全て閉じる．

クライアントからの接続があった場合　接続を受け付け，クライアントデータ構造を作成し，クライアントの構造体配列に追加する．

クライアントが購入商品が未選択の場合

- クライアントからの入力が `ls` の場合は，在庫一覧を送信する．
- クライアントからの入力が EOF の場合は，クライアントを切断し，クライアントデータ構造を削除する．
- クライアントからの入力が商品名の場合は，商品の在庫があるか確認し，在庫があれば，商品の構造体ポインタをクライアント構造体に設定する．

クライアントが購入商品を選択している場合

- クライアントからの入力が EOF の場合は，クライアントを切断し，クライアントデータ構造を

削除する．
- クライアントからの入力が数値の場合は，投入金額を更新する．投入金額次第で，購入，返金，投入金額の表示を行う．
- 購入後は，商品の在庫を減らし，クライアントが保持している商品構造体ポインタを NULL にする．

工夫点

グローバル変数の利用　グローバル変数の利用を最小限に抑えることで，意図せぬ変更を防ぐ．グローバル変数として定義されているものは， `Product` 構造体の配列， `Client` 構造体の配列，プロダクトの個数である `int` 型の変数である．

構造体のポインタの利用　構造体のポインタを利用することで，データの冗長化を防ぐ．商品構造体のポインタをクライアント構造体に持たせることで，購入情報の冗長化を防ぐ． **関数の分割** クライアントリスト，商品リストや名前から商品構造体を取得する関数など，関数を分割することで，可読性を向上させる．グローバル変数を極力用いない設計にしたため，関数の引数にはしばしばポインタを用いる．

考察

今回は，サーバからクライアントへの応答（商品リストなど）をテキストで行った．しかし，データ形式を JSON などにし，クライアント側で出力文を組み立てることで，ネットワークの負荷を軽減できると考える．

感想

本課題，またこの講義を通して，現代のネットワークインフラを支える技術とその危険性，また我々開発者が心得るべきことを学んだ．講義の内容は非常に興味深く, また実際にプログラムを書くことで，理解が深まった．教科書にはない，現状の課題や技術についても深く教えてくださった敷田教授，ありがとうございました．

溝口 洸熙

実行ファイルの生成と出力

Listing 3　実行ファイルの生成

```
$ make server
gcc -c server . c
gcc -c myio . c
gcc -o Server . out server . o myio .
   o
$ make client
```

Listing 4　サーバ側の出力

```
cls
>> Clients list :
| - Client [0]
| | - socket : 4
| | - ordered : none
|  ‘- payed : 0
| - Client [1]
| | - socket : 5
| | - ordered : none
|  ‘- payed : 0
ls
>> Products list to { fds : 1}
| - Product [ 0]: apple , price : 70 ,
left : 10
| - Product [ 1]: coffee , price : 100
    ,
left : 1
| - Product [ 2]: milk , price : 40 ,
left : 8
| - Product [ 3]: orange , price : 80
   ,
left : 12
| - Product [ 4]: tea , price : 50 ,
left : 15
```

Listing 5　クライアント側出力例

```
$ nc 127.0.0.1 10000
|- Product[ 0]: apple  , price:  70,
   left: 10
|- Product[ 1]: coffee , price: 100,
   left:  1
|- Product[ 2]: milk   , price:  40,
   left:  8
|- Product[ 3]: orange , price:  80,
   left: 12
|- Product[ 4]: tea    , price:  50,
   left: 15
apple
The price is 70 yen
20
apple : 70 yen , you payed 20 yen .
50 ¥ more , please
60
apple : 70 yen , you payed 80 yen .
Here is your change : 10 yen
Enjoy your drink !
ls
|- Product[ 0]: apple  , price:  70,
   left:  9
|- Product[ 1]: coffee , price: 100,
   left:  1
|- Product[ 2]: milk   , price:  40,
   left:  8
|- Product[ 3]: orange , price:  80,
   left: 12
|- Product[ 4]: tea    , price:  50,
   left: 15
```