応用プログラミング ex6 演習課題

1.積算クライアントの作成 <color white,red/white>[TA]</color>

- 積算クライアント addClient を作成しなさい。
- addClient は、echoClient を修正して作成します。これまでに改良してきた各自のプログラムを利用しても構いません。
- 演習時間中、教員用端末(10.10.113.181) で積算サーバを起動しておきます。起動ターミナルをプロジェクタスクリーンに表示しておきますので、動作確認に利用してください。
- %~~プログラムが完成したら□TA にクライアントの実行過程を示し、必要に応じてソースコードの変更点を説明しなさい。~~%

- &ref{add_fgets.c};

2.積算サーバの作成 <color white,red/white>[TA]</color>

- 積算サーバ addServer を作成しなさい。
- addServer は、echoServer を修正して作成します。これまでに改良してきた各自のプログラムを利用しても構いません。
- %~~プログラムが完成したら、1.で作成したクライアントを用いて積算サーバの実行の様子を示し、必要に応じてソースコードの変更点を説明しなさい。~~%

- &ref{add_strtok.c};

3.積算サーバ／クライアントの改良

- 余裕のある人はチャレンジして下さい。
- 作成した addServer/addClient に以下のような機能を追加しなさい。
- 必要に応じて積算プロトコルの仕様を修正してもかまいません。（例えば、サーバからの返信バイト数を 64 バイト未満に制限するなど）

1. **積算値をゼロにリセットできるようにする。**
    - 特定の文字列（例えば□”clear“□をサーバに送信すると、サーバ上の積算値が 0.0 にリセットされる。
2. **addClient から繰り返し積算データを送信できるようにする。**

//—addServer から返信される総計は、送信ごとの総計、あるいは一度の接続における積算のどちらでもかまいません。

- 繰り返し入力を終了する方法を含むこと。（例えば□”quit“ の入力で終了するなど）

1. **addServer でトークンの分割エラー（トークンがひとつも見つからない）があれば、合計ではなくエラー文字列を返信する。**

- 送信文字列にトークンがひとつも見つからない場合、数値への変換も、それらを合計することもできません。
- strtok() の返り値を調べるとトークンの有無がわかります。

1. **addServer が atof() で変換できない文字列を受け取ったら、合計ではなくエラー文字列を送信**

する。（やや難しい）

- atof() の代わりに、同じ <stdlib.h> の関数 strtod() を利用すると、変換と同時に変換できなかった文字列を調べることができます。

1. **addServer の受信メッセージが長過ぎたら、合計ではなくエラー文字列を送信する。（難しい）**

- 先の例では□addClient が送信文字列を制限していましたが、必ずしも送信側のクライアントプログラムが文字数制限(1-63)を守っている保証はありません□addServer 側で文字列が制限を超えていないかをチェックしてください。
- addServer は一時的に文字数制限を超える文字列データを受け取れる必要があります。また、文字数制限を超えた文字列データを空読みして捨てる必要があります。捨てるべきデータがあるかどうかは select() 関数などで調べることができます。

//-%実行結果、~~および改良した点とその実装方法を TA に説明しなさい。~~%

- 以下は実行結果の一例です。

[sano@s1542f160 ~]$ ./addClient 127.0.0.1 5007 *Variables

```
Server IP: 127.0.0.1
Server Port: 5007
```

*Socket Ready *Server Connected Values? 1.2 3e-1 □□□□□□□□□□← 普通に計算 *String Length: 8 *Sent String: 1.2 3e-1 Received: 1.500000 Values? clear □□□□□□□□□□ ← 0 にリセット *String Length: 5 *Sent String: clear Received: 0.000000 Values? 1.2 3e-1 □□□□□□□□□□← 普通に計算 *String Length: 8 *Sent String: 1.2 3e-1 Received: 1.500000 Values? 4.5 ← 終了せずに次を入力 *String Length: 3 *Sent String: 4.5 Received: 6.000000 ← 積算した結果が返る Values? 100000000 200000000 300000000 400000000 500000000 600000000 700000000 ← 63バイトを超えた送信 *String Length: 69 *Sent String: 100000000 200000000 300000000 400000000 500000000 600000000 700000000 Received: Error: too long data line. (6.000000) ← 長過ぎるのでエラー（積算も返す） Values? ← 空白文字を3バイト *String Length: 3 *Sent String: Received: Error: can't find token. (6.000000) ← トークンが無いのでエラー（積算も返す） Values? 6.7 abc.d *String Length: 9 *Sent String: 6.7 abc.d ← 2つ目のトークンが変（数値ではない□ Received: Error: invalid string: “abc.d”. (12.700000) ← 数値に変換できないのでエラー（数値 6.7 は加算している□ Values? quit □□ ← “quit” を入力 *String Length: 1 *Sent String: 0 □ ← 内部で “ 0 ” を送信 Received: 12.700000 □ ← +0 の結果を返して接続終了 *Close Connection [sano@s1542f160 ~]$ }}