応用プログラミング ex4 演習課題

1.いろいろな型の sizeof <color white,red/white>[TA]</color>

- 以下に示す型、および変数のサイズ（バイト数）を sizeof 演算子を使って調べなさい。
- **型**
  - int
  - short int
  - unsigned int
  - long int
  - long long int
  - double
  - long double
  - char
  - struct str {
    char c;
    int a;
    double x;
    };
- **変数**
  - unsigned int a[10];
  - long double x[10];
  - struct str tmp[10];
  - char s[] = “123456”;
  - int *ip;
  - double *dp;
- %~~上に宣言された6つの変数それぞれのサイズ（バイト数）を sizeof 演算子を使って調べ、なぜそのような値になるのかを考えて TA に説明しなさい。~~%

2.echoClient で送信文字列をキーボードから入力し、繰り返し送信できるようにする。

2-1.送信文字列の設定をコマンドライン引数から、キーボード入力へと変更する<color white,red/white>[TA]</color>

1. 送信文字列用 echoString の宣言時領域確保（最大送信文字数の定義 (echo.h)
   - **ポインタとして宣言されている** echoString **を、具体的な大きさをもった文字配列(char [])に変更し**、送信文字列用**のメモリ**領域を確保します。
   - echoString の大きさは、1回に送信する文字列を格納するのに十分な大きさを持ったものとし、**適当な名前で echo.h に定数定義して**利用します。
2. 引数処理の変更
   - **コマンドライン**引数からの送信文字列の代入を停止します。その際、必要とされる**コマンドライン引数の数が減る**ので、それに合わせてコマンドライン引数の処理も変更してください。
3. 送信文字列をキーボードから入力する
   - scanf() あるいは fgets() などを利用して、キーボードから echoString に送信文字列を入力

します□fgets() でキーボード入力する場合は、最後の引数を（標準入力を表す）stdin とします。
    - 空白を含めた文字列を処理したい場合は、fgets() を利用する方が簡単です□fgets() は前期の馬さんの講義（プログラミング・演習）で解説されています。
    - fgets()を用いた場合、入力には改行文字も含まれます。この入力をそのまま利用すると改行文字もサーバへ送信されてしまいますが、ここではそのままでも構いません。
    - キーボード入力が最大送信文字列を超えた場合の処理については、**ここでは考えなくて構いません**□
4. echoServer と通信してキーボードからの入力が送信されていることが確認しましょう。
%~~動作が確認できたら TA に通信の様子を示し、プログラム上で変更した箇所を説明してください。~~%

2-2.サーバ接続を維持したまま、複数回繰り返し echoServer との通信を行う。<color white,red/white>[TA]</color>

//-余裕のある人はチャレンジしてみてください。

- **サーバとの接続確立後の**「送信文字列のキーボード入力・送信・受信」の一連の通信手順を繰り返し、連続してサーバとの通信ができるようにします。
- 送信文字列入力が **quit** であれば、ループを終了して（入力された**quit**は送信せずに）接続を切断する。文字列の比較には//strcmp(// を使います。
- 実行結果の一例を示します。（以下の例は、前回課題の状態表示を含んでいます。）

```
[sano@s1542f160 ~]$ ./echoClient 127.0.0.1 5007
*Variables
  Server IP: 127.0.0.1
  Server Port: 5007
*Socket Ready
*Server Connected
echo String? ABC
*echo Length: 3
*Sent String: ABC
Received: ABC
echo String? 1234567
*echo Length: 7
*Sent String: 1234567
Received: 1234567
echo String? quit
*Close Connection
[sano@s1542f160 ~]$
```

1. %~~動作が確認できたら TA に通信の様子を示し、プログラム上で変更した箇所を説明してください。~~%

2-3.送信文字入力での文字数制限（キーボード入力が最大送信文字数を超えた場合の処理）

- 余裕のある人はチャレンジしてみてください。

= scanf() 入力のための情報：

- scanf() の文字列代入(%s)では、

    1. 先頭の空白、改行は読み飛ばされます。
    2. 空白、改行が来ると代入が終了します。
- 以下の例では//"%63s%*[^\n]// のような書式が使われています。

1. //%63// は、最大63文字を文字列として読むことを指定します。
2. //%*[^\n// は、次のものの組合わせです。

- //%[文字集合// は、[]内に含まれる文字だけを文字列とします。
- //%[^文字集合// は、[]内に含まれない文字だけを文字列とします。
- //%*指定// は（指定子は代入の型を指示する %d,%f,%s などのこと）、入力を読むが変数への代入は行わず捨てます。

= fgets() 入力のための情報:

- fgets() は、\n まで、あるいは（指定最大文字数-1）文字まで読み、最後に \0 を 付けます。
- 入力が（指定最大文字数-1）以下で \n を含んでしまう場合には□\n を \0 に 置き換える必要があります。この処理を行わない場合□\n もサーバへ送信されてしまいます。

strchr(echoString, '\n'); }} などが使えるでしょう。

- 入力が（指定最大文字数-1）を超える場合には、残った入力をクリア（空読み）しておかないと、次回の入力時に残ってしまいます。

while(getchar() != '\n') {...} }} などが使えるでしょう。

- scanf() を使った場合の実行例

```
[sano@s1542f160 ~]$ ./echoClient 127.0.0.1 5007
*Variables
  Server IP: 127.0.0.1
  Server Port: 5007
*Socket Ready
*Server Connected
echo String? 1234567
*echo Length: 7
*Sent String: 1234567
Received: 1234567
echo String? 123 456       // 空白を含む入力
*echo Length: 3
*Sent String: 123          // 空白以降は捨てられる
Received: 123
echo String?
.........*.........*.........*.........*.........*.........*.........*  //
70文字入力
*echo Length: 63
*Sent String:
.........*.........*.........*.........*.........*.........*...        // 超
えた分は捨てられる
Received: .........*.........*.........*.........*.........*.........*...
echo String? quit
*Close Connection
[sano@s1542f160 ~]$
```

- fgets() を使った場合の実行例

```
[sano@s1542f160 ~]$ ./echoClient 127.0.0.1 5007
*Variables
  Server IP: 127.0.0.1
  Server Port: 5007
*Socket Ready
*Server Connected
echo String? 1234567
*echo Length: 7
*Sent String: 1234567
Received: 1234567
echo String? 123 456       // 空白を含む入力
*echo Length: 7
*Sent String: 123 456      // 空白も含んで送信
Received: 123 456
echo String?
.........*.........*.........*.........*.........*.........*.........*  //
70文字入力
*echo Length: 63
*Sent String:
.........*.........*.........*.........*.........*.........*...        // 超
えた分は捨てられる
Received: .........*.........*.........*.........*.........*.........*...
echo String? quit
*Close Connection
[sano@s1542f160 ~]$
```