応用プログラミング ex5 演習課題

1.echoServer で実行状況の出力表示 <color white,red/white>[TA]</color>

- コメント付きソースファイル
  - &ref{echoServer_wc.c};
  - &ref{handleTCPClient_wc.c};
- 次の各ポイントで、**echoServer の状態を表示するように** echoServer.c および handleTCPClient.c を変更しなさい。その際、**各ポイントで echoServer の実行を1秒づつ休止しなさい**。

1. **サーバポートの**確定後
2. 着信**ソケット**作成後
3. 着信**ソケットのバインド**後
4. 着信**ソケットを**接続待機状態(Listen)に設定後
5. **クライアントから**着信接続を待機する前
6. **クライアントから**着信接続を受け入れ後
7. **クライアントからデータ**受信後
8. **クライアントへデータ**送信後
9. **サーバ**接続切断後

- **プログラムを**途中で休止**するためには** unistd.h で定義**されている**関数 sleep() を利用**しなさい**。
- **クライアントから**送**られてくる**文字列**には、**終端文字 '\0' **が付いていません。そのため、**受信データ（文字列）を printf() の文字列(%s)書式**などでそのまま**表示しようとするとうまく表示**されません** handleTCPClient.c **のように受信データの長さを使って %c で表示するなどの**工夫**が必要です**。送信時**についても**同様**です**。
- 出力**されたはずの**状態表示**がすぐには**表示**されない**場合**があります。これは、**表示速度**などの**効率化**のために、ある程度まとまったサイズになるまで実際の表示出力を貯めておくしくみ（バッファ）があるためです**。表示**がすぐになされない**場合**は、**stdio.h で定義**されている**

fflush(stdout); }} という、貯まっているバッファを強制的に吐き出す（表示させる）関数を printf() などの直後に挿入してみてください。

- **サーバのIPアドレスは** INADDR_ANY **で**自動取得**するため、**状態表示**で表示する必要はありません。**以下**の例**では、**関数 char *inet_ntoa(struct in_addr in) **を**利用**しています**。

printf("*Bind to %s:%u\n", inet_ntoa(echoServAddr.sin_addr),... }}

- 実行結果**の**一例**を**示す。

```
[sano@s1542f160 ~]$ ./echoServer 5007
*Server Port: 5007                      <- ポート番号表示して、1秒休止
*Socket Ready                           <- ソケット作成して、1秒休止
*Bind to 0.0.0.0:5007                   <- ソケットをバインドして、1秒休止（サーバIPは INADDR_ANY で自動で取得させるため 0.0.0.0 と表示される）
*Server socket to be Listen             <- ソケットを待機状態にして、1秒休止
*Waiting Client connection.             <- クライアントからの接続を待つ
*Accept Client connection.              <- クライアントから1度目の接続を受け取って、
```

```
1秒休止
Handling client 127.0.0.1                <- クライアントIP表示（元からの出力）
*Received Data: 1st connection           <- 受信したデータを表示（元からの出力）、1秒
休止
*Sent Data: 1st connection               <- 送信したデータを表示、1秒休止
*Received Data: 12345
*Sent Data: 12345
*Received Data: .........*.........*.........*..    <- 受信バッファサイズを超えた
データ（50バイト）は2回に分けて受信／送信している（初回、32バイト）
*Sent Data: .........*.........*.........*..
*Received Data: .......*.........*                   <-□二回目、残り18バイト）
*Sent Data: .......*.........*
*Received Data:
*Close Connection                                    <- クライアントの接続が切れた
*Waiting Client connection.                          <- 次のクライアントからの接続
を待つ
*Accept Client connection.                           <- 2回目の接続を受け取った
Handling client 127.0.0.1
*Received Data: 2nd Connection
*Sent Data: 2nd Connection
*Received Data:
*Close Connection                                    <- 2回目の接続を終了して
*Waiting Client connection.                          <- 次の接続を待つ
```

- %~~必要に応じてプログラムソースを参照しながら、実行結果を TA に説明しなさい。~~%

2.echoServerから1文字ずつの返信 <color white,red/white>[TA]</color>

- echoServer の**クライアントへの返信を1文字(1byte)ずつ**行うように変更しなさい。その際、**1文字返信するごとにechoServerのプログラム実行を1秒間休止**しなさい。
- 演習課題 1.で行った送信データの表示についても、**1秒毎の送信に合わせて1文字ずつ表示すること**。
- **クライアントからの受信データ方法は変更せずにまとめて受信し、表示もまとめて表示します。**
- **echoClient.c を変更する必要はありません。ただし**□echoClient 側でも1文字ずつの受信を確認したい場合には□echoClient.cで毎回の受信時の受信文字列表示後にfflush(stdout);を追加します。
- %~~変更した箇所を説明しながら、実行結果を TA に示しなさい。~~%