ソフトウェア設計及び実験
統合・モジュール化

担当：吉田

2019 年 10 月 15 日

1　はじめに

通常，プログラムを作成する場合，まず，データ構造･モジュール･モジュール内の関数･（ネットワークを使うプログラムでは）コマンドプロトコルを設計してから，実際のプログラミングを開始する．大規模なプログラムを作成する場合や，複数人で作成する場合には，この設計が特に重要となる．設計が悪いとプログラム作成中に不都合が生じたり，複数のソースファイルを組み合わせた時にリンクエラーが起こるなどして，設計のやり直しや関数名・変数名の変更などに時間をとられ，開発が遅れることになる．逆に，設計を綿密にしておけば，開発が比較的容易に進み，ソースファイルのリンクもスムーズに行うことができる．

そこで，本実験では，サンプルプログラムを用いてプログラムの設計方法について学ぶ．

2　プログラム設計

2.1　モジュール化

プログラムをモジュールにわける時，まずしなければならないのは，プログラムで行わなければいけない処理を全てリストアップすることである．次に，それぞれの処理の関連性を調べて，関連性の高い機能を集めて，一つのモジュールとする．

今回の実験のように，SDL による GUI とネットワークを使ったゲームを作成する場合，大きく以下のモジュールに分けることができる．

1. サーバー
    - ネットワーク
    - コマンド処理

- ゲーム制御

2. クライアント

   - ネットワーク
   - コマンド処理
   - ユーザインターフェース
   - ゲーム制御

ネットワークモジュールは，サーバークライアント間でデータの送受信を行うモジュールである．コマンド処理モジュールはネットワークで送られてきたコマンドと引き数から実際に行う処理を決定したり，ユーザーの行動などをネットワークで送信するためのデータ（コマンド＋引き数）に変換するモジュールで，ユーザインターフェースモジュールは，SDL による GUI を用いてユーザーと対話を行うモジュールである．ゲーム制御モジュールは，ゲームのメイン部分になり，得点管理等を行う．これらのモジュールはさらに細かくわけることができる．(UFO モジュールとか，弾モジュールなど)

通常，1 つのモジュールは 1 つのファイルに記述し，1 人の人が作成する．

2.2 データ構造と定数

まず，設計しておかなければならないデータ構造は，複数のモジュールで使用する構造体や，マクロ（定数）などである．モジュール内でのみ使用する構造体やマクロは，他のモジュールに知らせる必要がない（ブラックボックス）ので，モジュール内の設計を行う時に決めれば良い．本実験の場合には，まず，サーバーとクライアントで共通なデータ構造を決定し，次に，サーバーとクライアントそれぞれについて，モジュール間で受け渡しを行う構造体や，共通なマクロを定義する．ここで決めた構造体とマクロは，モジュール共通のヘッダーファイルで定義しておかなければならない．

2.3 外部関数

外部関数とは，各モジュールが外部に公開する関数で，他のモジュールから呼ばれる関数である．関数名，機能，引き数，返り値を細かく決めておく必要がある．ここで決めた関数は，モジュール共通のヘッダーファイルでプロトタイプ宣言する必要がある．

関数呼び出し関係（コールグラフ）を書くことにより，各モジュールの関係と，作成しなければならない外部関数を明確にすることができる．

2.4　コマンドプロトコル

コマンドプロトコルとは，サーバーとクライアント間で送受信するデータの取り決めのことを言う．当然のことであるが，サーバーとクライアントの両方が送受信するデータについて知らなくてはならない．このデータで決めておかなければならないことは，コマンド，コマンドの引き数の意味（何を），送信元（だれが），送信先（だれに），送信タイミング（いつ），送信目的（何のために）である．

3　スレッド

3.1　マルチスレッドとは

一つのプログラム内で，複数の処理（スレッド）を並列に実行することをマルチスレッドと呼ぶ．マルチスレッドを用いれば，複雑な時間のかかる処理を行いながら，SDL のイベントを処理したり，ネットワークの送受信処理を行ったりすることができる．また，スレッドを用いれば，モジュールのきり分けが簡単になり，プログラムを効率的に設計することができる．詳しくは参考文献 [1] を参照して欲しい．

3.2　同期処理

複数のスレッドが，同じデータ（変数，ファイル，メモリなど）に同時にアクセスしてデータを破壊しないようにする処理を同期処理と呼ぶ．

ミューテックスを使用すれば，アクセスしたいデータにロックをかけることができ，ロックがかかっている間は，そのデータに他のスレッドはアクセスすることができない．

マルチスレッドプログラムでは，グローバル変数は全てのスレッドで同じ値となるので，スレッド間の同期を取るために使用できる．しかし，グローバル変数の値が変わるまで待ち続けるだけのループは，処理のオーバーヘッドが大きいので，使用は避けた方が良い．

4　課題

次のような仕様の GUI とネットワークを使った「じゃんけんゲーム」を作成しなさい．

クライアントの仕様：

- 画面に「グー」「チョキ」「パー」を表すボタンと，ゲームを終了させるボタンを表示する．
- 押されたボタンの情報をサーバーに送る．
- ボタンを押した後，サーバーから結果が送られてくるまで，ボタンを押してもサーバーにデータを送らない．
- サーバーから送られてきたデータが勝ち負けに関するものであれば，じゃんけんの結果を表示する．
- サーバーから送られてきたデータがゲームの終了を示すものであれば，クライアントを正常終了させる．
- 勝敗は printf 等の関数で表示するのではなく，SDL 関数を用いてウインドウ上に表示する．

サーバの仕様：

- ゲームに参加できるクライアントの数は 2 とする．
- 2 つのクライアントから送られたデータが，じゃんけんに関するもの（「グー」「チョキ」「パー」のいずれか）であれば，勝敗を判定し，その結果を両クライアントに送る．
- クライアントから終了を示すデータが送られてきた時，全クライアントにゲームの終了を示すデータを送り，サーバを正常終了させる．

余力のある人用（ボーナスポイント）：

- プログラムを起動してからの勝敗結果（何勝何敗何引き分けか）を画面上に表示する．
- ゲームに参加できるクライアントの数を 3 以上にする．
- その他，何でもよいので改良する．

これらの改良をした人は，改良ポイントをレポートに明記すること．

5　レポート

作成したゲームの内容，操作方法，モジュール，外部関数の説明，コマンドプロトコル（余裕があればコールグラフも）を記述しなさい（付録A．1節～A．5節を参考にすること）．図や画面のコピーを用いるなどして，分か

りやすく説明すること．これらが記述されていないレポートは必ず再レポートになります．ただし，外部関数とコマンドプロトコルは自分で作成したものや変更したものだけでよい．

- サンプルプログラムは，manaba から各自コピーすること.（スレッド有りと無しの 2 種類）
- manaba を使って，提出すること．
- レポートの PDF ファイルと．Makefile，全プログラムのソースファイル（オブジェクトファイル，実行ファイルは除く），画像ファイルなどを自分の学籍番号と同じ名前のディレクトリに入れ，そのディレクトリを圧縮（拡張子は tar.gz または tgz）して，提出すること．
  （例：tar zcvf c50xxxxxx.tar.gz c50xxxxxx)
- プログラムは，コンパイル，リンク，実行をするために必要なファイル全てを提出すること．
- logout する前に自分のプロセスを調べて，暴走しているプロセスがあれば削除する．

レポートの〆切は 10 月 21 日 (月)12 時 50 分
質問は吉田（mino@is.tokushima-u.ac.jp、エコ棟 704）まで。

参考文献

[1] Bradford Nichols, Dick Buttlar, Jacqueline Proulx Farrell, 榊正憲訳：Pthreads プログラミング , オライリー・ジャパン (1998).

A　サンプルプログラム

ここでは，サンプルプログラムを作成する際に設計したことを記述してあります．自分でプログラムを設計する際の参考にして下さい．

A.1　プログラム内容

このプログラムは，SDL による GUI と，ネットワークを使った簡単なサンプルプログラムです．ユーザーがボタンを押すことにより，自分自身や他のユーザーの画面に円や四角を表示することができます．また，５秒毎に菱形を表示します．円・四角・菱形を表示させる位置と大きさは，乱数を用いて決定しています．

A.2　起動方法

まず，サーバーを起動し，その後，サーバー起動時に指定した数のクライアントを起動させます．

A.2.1　サーバーの起動方法

`server` クライアント数

クライアント数は，1〜5 の値を指定できます．

A.2.2　クライアントの起動方法

`client` ホスト名

サーバーが起動しているホスト名を引き数として指定します．サーバーと接続後，ユーザーの名前を入力します．

A.3　操作方法

画面上部に表示されているボタン，ユーザーの名前が表示されているボタンを押すと，そのクライアントの画面上に円が一つ表示されます.「All」と表示されているボタンを押すと，全クライアントの画面の同じ場所に四角が一つ表示されます.「End」と表示されているボタンを押すと，プログラムが終了します．

A.4　モジュール

A.4.1　サーバー

サーバーのモジュールの構成を図 1 に示す.

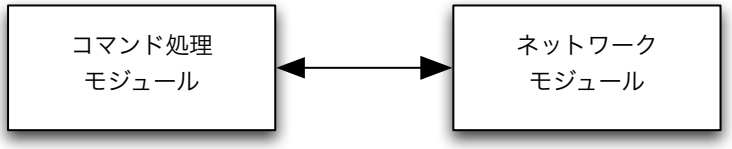

図 1: サーバーのモジュール構成

以下に，各モジュールの外部関数を説明する.

1. ネットワークモジュール

   - int SetUpServer(int num)
     機能：クライアントとのコネクションを設立し，ユーザーの名前の送受信を行う
     引き数：クライアント数
     出力：コネクションに失敗した時-1, 成功した時 0
   - void Ending(void)
     機能：全クライアントとのコネクションを切断する
     引き数：なし
     出力：なし
   - int RecvIntData(int pos,int *intData)
     機能：クライアントから int 型のデータを受け取る
     引き数 1：クライアント番号
     引き数 2：受信したデータ
     出力：受け取ったバイト数
   - void SendData(int pos,void *data,int dataSize)
     機能：クライアントにデータを送る
     引き数 1：クライアント番号
     引き数 2：送るデータ
     引き数 3：送るデータのサイズ
     出力：なし
   - int SendRecvManager(void)
     機能：クライアントから送られてきたデータを処理する
     引き数：なし

出力：プログラム終了コマンドが送られてきた時 0 を返す．それ以外は 1 を返す．

2. コマンド処理モジュール

- int ExecuteCommand(char command,int pos)
  機能：クライアントから送られてきたコマンドを元に，引き数を受信し，実行する
  引き数 1：コマンド
  引き数 2：クライアント番号
  出力：プログラム終了コマンドがおくられてきた時には 0 を返す．それ以外は 1 を返す
- void SendDiamondCommand(void)
  機能：クライアントに菱形を表示させるためにデータを送る
  引き数：なし
  出力：なし

A.4.2 クライアント

クライアントのモジュールの構成を図 2 に示す．

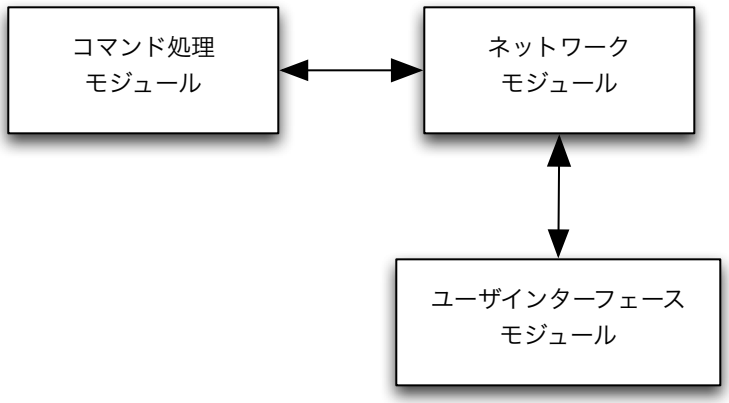

図 2: クライアントのモジュール構成

以下に，各モジュールの外部関数を説明する．

1. ネットワークモジュール

- int SetUpClient(char* hostName,int *num,char clientName[][])
  機能：サーバーとのコネクションを設立し，ユーザーの名前の送受信を行う

引き数 1：ホスト名
引き数 2：全クライアント数
引き数 3：全クライアントのユーザー名
出力：コネクションに失敗した時-1, 成功した時 0

- void CloseSoc(void)
  機能：サーバーとのコネクションを切断する
  引き数：なし
  出力：なし
- int RecvIntData(int *intData)
  機能：サーバーから int 型のデータを受け取る
  引き数 1：受信したたデータ
  出力：受け取ったバイト数
- void SendData(void *data,int dataSize)
  機能：サーバーにデータを送る
  引き数 1：送るデータ
  引き数 2：送るデータのサイズ
  出力：なし
- int SendRecvManager(void)
  機能：サーバーから送られてきたデータを処理する
  引き数：なし
  出力：プログラム終了コマンドが送られてきた時 0 を返す．それ以外は 1 を返す.

2. ユーザインタフェースモジュール

- void InitWindows(int num,char name[][])
  機能：メインウインドウの表示，設定を行う
  引き数 1：全クライアント数
  引き数 2：全クライアントのユーザー名
  出力：なし
- void DestroyWindow(void)
  機能：メインウインドウを破棄して，SDL を終了する
  引き数：なし
  出力：なし
- void WindowEvent(int num,char name[][])
  機能：メインウインドウに対するイベント処理を行う
  引き数 1：全クライアント数
  引き数 2：全クライアントのユーザー名
  出力：なし

- void DrawRectangle(int x,int y,int width,int height)
  機能：メインウインドウに四角を表示する
  引き数 1：四角の左上の x 座標
  引き数 2：四角の左上の y 座標
  引き数 3：四角の横幅
  引き数 4：四角の高さ
  出力：なし
- void DrawCircle(int x,int y,int tyokkei)
  機能：メインウインドウに円を表示する
  引き数 1：円の中心の x 座標
  引き数 2：円の中心の y 座標
  引き数 3：円の直径
  出力：なし
- void DrawDiamond(int x,int y,int height)
  機能：メインウインドウに菱形を表示する
  引き数 1：菱形の中心の x 座標
  引き数 2：菱形の中心の y 座標
  引き数 3：菱形の高さ
  出力：なし

3. コマンド処理モジュール

- int ExecuteCommand(char command)
  機能：サーバーから送られてきたコマンドを元に，引き数を受信し，実行する
  引き数：コマンド
  出力：プログラム終了コマンドがおくられてきた時には 0 を返す. それ以外は 1 を返す
- void SendRectangleCommand(void)
  機能：クライアントに四角を表示させるために，サーバーにデータを送る
  引き数：なし
  出力：なし
- void SendCircleCommand(int pos)
  機能：クライアントに円を表示させるために，サーバーにデータを送る
  引き数：円を表示させるクライアントの番号
  出力：なし
- void SendEndCommand(void)
  機能：プログラムの終了を知らせるために，サーバーにデータを

送る
引き数：なし
出力：なし

A.5 コマンドプロトコル

- コマンド (なし)
  引き数：クライアントのユーザーの名前 (char * 10)
  クライアントからサーバーへ
  クライアントが起動した直後
  ユーザー名をサーバーに知らせるため

- コマンド (なし)
  引き数 1：クライアント数 (int)
  引き数 2：全てのユーザーの名前 (char * 10 * クライアント数)
  サーバーから全クライアントへ
  全てのクライアントからユーザー名が送られてきた時
  全ユーザー名を全クライアントに知らせるため

- コマンド (C)
  引き数：円を表示するクライアント番号 (int)
  クライアントからサーバーへ
  ユーザーがユーザー名の書かれたボタンを押した時
  サーバーにボタンが押されたことを知らせるため

- コマンド (C)
  引き数 1：円の x 座標 (int)
  引き数 2：円の y 座標 (int)
  引き数 3：円の直径 (int)
  サーバーからクライアントへ
  クライアントからコマンド (c) が送られてきた時
  クライアントの画面に円を表示させるため

- コマンド (R)
  引き数：なし
  クライアントからサーバーへ
  ユーザーが「All」と書かれたボタンを押した時
  サーバーにボタンが押されたことを知らせるため

- コマンド (R)
  引き数 1：四角の左上の x 座標 (int)
  引き数 2：四角の左上の y 座標 (int)

引き数 3 : 四角の幅 (int)
引き数 4 : 四角の高さ (int)
サーバーから全クライアントへ
クライアントからコマンド (R) が送られてきた時
全クライアントの画面に四角を表示させるため

- コマンド (D)
  引き数 1 : 菱形の中心の x 座標 (int)
  引き数 2 : 菱形の中心の y 座標 (int)
  引き数 3 : 菱形の高さ (int)
  サーバーから全クライアントへ
  ５秒間に 1 回（割り込み処理）
  全クライアントの画面に菱形を表示させるため

- コマンド (E)
  引き数 : なし
  クライアントからサーバーへ
  ユーザーが「End」と書かれたボタンを押した時
  サーバーに終了ボタンが押されたことを知らせるため

- コマンド (E)
  引き数 : なし
  サーバーから全クライアントへ
  クライアントからコマンド (E) が送られてきた時
  プログラムを終了させるため