

# プログラミング演習Ⅲ（テーマ1）

## 第6回

- Virtual BoxでCentOSを起動し、ログインしておくこと。
- **Manaba**にて下記資料をダウンロード
  - 第6回講義資料(part6.pdf)
  - 技術文書の書き方
  - **レポート表紙**（(学籍番号)プロ演3最終レポート表紙.docx）
  - **評価用プログラムの実行ファイル**（4moku\_sample）

# 13. 最終テーマ

(X11とソケット通信を用いたイベント駆動型プログラミング)

- インターネットにおけるホスト名 + ドメイン名の「ホストコンピュータ識別子」を使用して対戦相手を指定する／されることにより、対戦相手どうしの間の通信路を確立したうえで「ネットワーク対戦型四目並べゲーム」を行えるプログラムを作成すること.
- この際、同プログラムはTCP/IPプロトコル上で指定されたアプリケーションプロトコルを実装すること. 従ってX11およびTCP/IPソケットを用いて作成すること.
- また、ユーザが自ら対戦相手を指定して接続要求を出すか (クライアント)、対戦相手から接続要求があるまで待機する (サーバ)かを選択できるようにプログラムを設計すること.
- ポート番号は十進数の20,000を用いること.

# 13. 最終テーマ

□テーマの仕様: 以下の仕様を必ず満たすこと.

- ①動作環境:「プログラム演習3テーマ1演習環境準備」マニュアルによって構築した CentOS
- ②開発環境: TCP/IPソケットライブラリ+X11ライブラリ, GNU C コンパイラ  
上記動作環境の端末において「**gcc -lX11**」でコンパイルできること  
#複数プログラムファイルで構築する場合はコンパイルコマンドについて  
#必ず明記すること 例: `gcc -lX11 pro1.c pro2.c pro3.c`
- ③碁盤サイズ: 縦6マス × 横7マス
- ④通信方式: TCP/IPのソケット(コネクション型)を用いること. アプリケーション  
プロトコルとして, 相手のコンピュータとの通信には次ページの3つのメッセージを  
定義する. これ以外のメッセージを受信した場合は無視すること.

※アプリケーションプロトコルは本来自分達で設計しても構わないが, 今回の  
テーマでは全員がそれぞれ対戦できるようにするため, 予めアプリケーション  
プロトコルを指定している.

# 13. 最終テーマ

□テーマの仕様: 以下の仕様を必ず満たすこと

表: アプリケーションプロトコルのメッセージタイプ

メッセージ	文字コード	解釈
メッセージ①	"PLACE-XY"	<ul style="list-style-type: none"><li>・(X, Y)に基石を置いた際に送信</li><li>・X, Yはそれぞれ16進数の"0~6"の1桁</li><li>・(例)[0, 6]に置いた場合⇒"PLACE-06"</li><li>・接続後, サーバ⇒クライアントの順で交互に送信</li></ul>
メッセージ②	"ERROR"	<ul style="list-style-type: none"><li>・メッセージ①を受信後、下記の場合に送信 (既に基石が置いてある場所に<u>相手が</u>基石を置こうとしている) (<u>相手が</u>連続して基石を置こうとしている) (<u>相手が</u>下のマスに基石がない場所に基石を置こうとしている)</li><li>・全碁盤目が埋まっても勝負がつかない際に送信 (42手目を打った時点で"ERROR"を返す)</li><li>・ERRORを受信した側が接続を切断(close)</li></ul>
メッセージ③	"YOU-WIN"	<ul style="list-style-type: none"><li>・相手が4つ目を並べた際に送信. つまり, メッセージ①を受け取った際に4目並べが成立したか否かを調べ, 成立した場合にはこのメッセージを送信する. このメッセージを受信した側は接続を切断(close)する.</li></ul>

上記の解釈で不明な点がある場合はサンプルプログラムで動作を必ず確認すること

# 13. 最終テーマ

□テーマの仕様: 以下の仕様を必ず満たすこと.

## ⑤ルール

- ・碁石はサーバが先手(黒石), クライアントが後手(白石)で交互に置いていく
- ・既に碁石が置かれている場所に新たに碁石を置くことはできない
- ・下のマスに碁石がないマスには碁石を置くことはできない
- ・下のマスから碁石を順に積み重ねていく
- ・縦、横、斜め、いずれかの方向に同色の碁石を先に四目並べた方が勝ち
- ・盤が碁石で埋まっても勝負がつかない場合は引き分けとする  
(この場合最後の一手を打つPLACE\_XYメッセージが送信され、これを受けた側がERRORメッセージを送信し、これを受けた側からcloseを発信し終局される)

※ルールについては, 下記を参照すると良い.

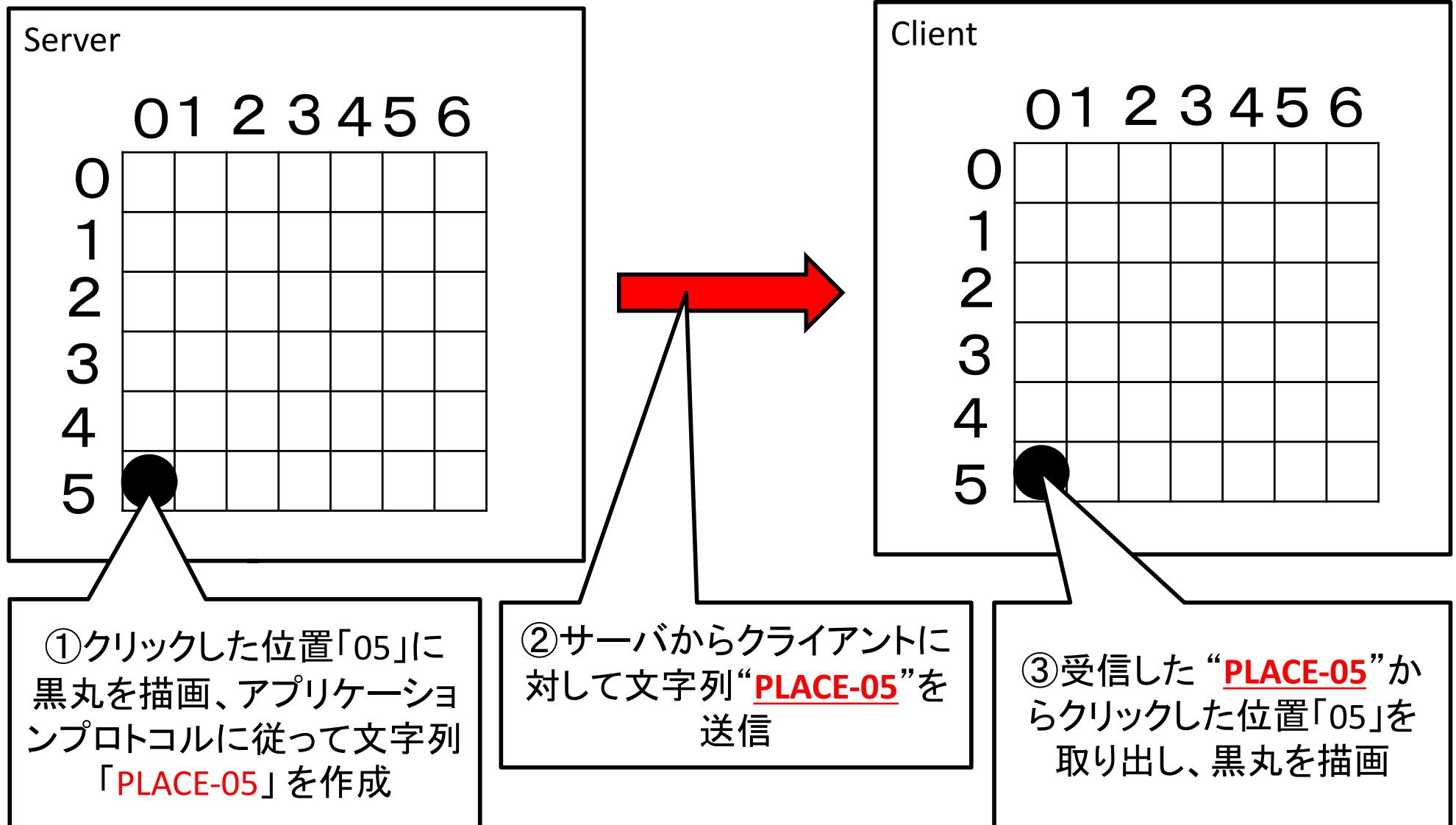
[http://www.daiichi-g.co.jp/osusume/forfun/06\\_yonmoku/06.html](http://www.daiichi-g.co.jp/osusume/forfun/06_yonmoku/06.html)

## ⑥再描画処理

- ・ウィンドウが他のウィンドウに遮蔽された場合、ウィンドウサイズを変更した場合、それまでの履歴にしたがって置かれた碁石をすべて再描画すること

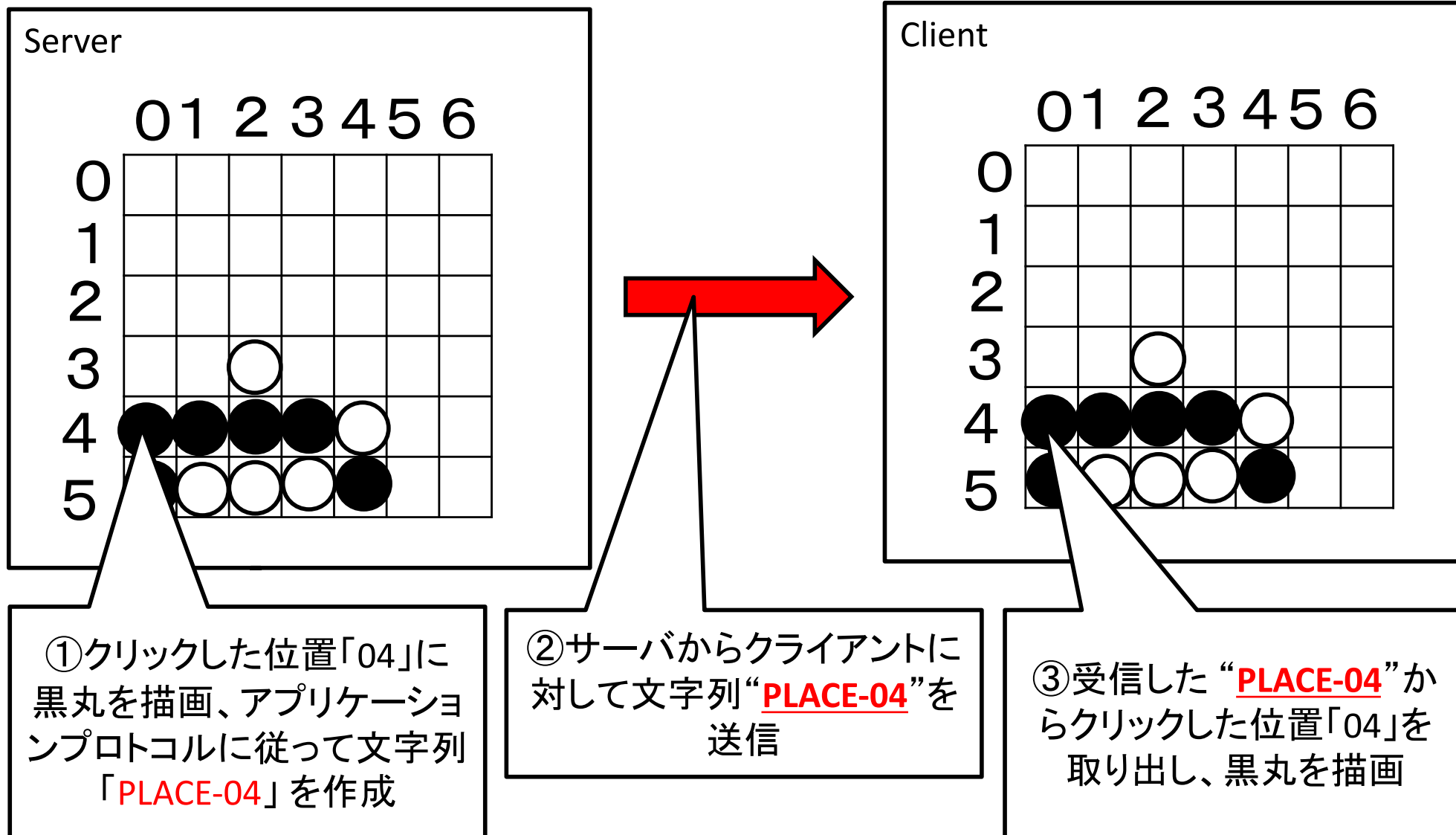
# 13. 最終テーマ

□ 基石を置いた際の基石描画までのプログラムの流れ



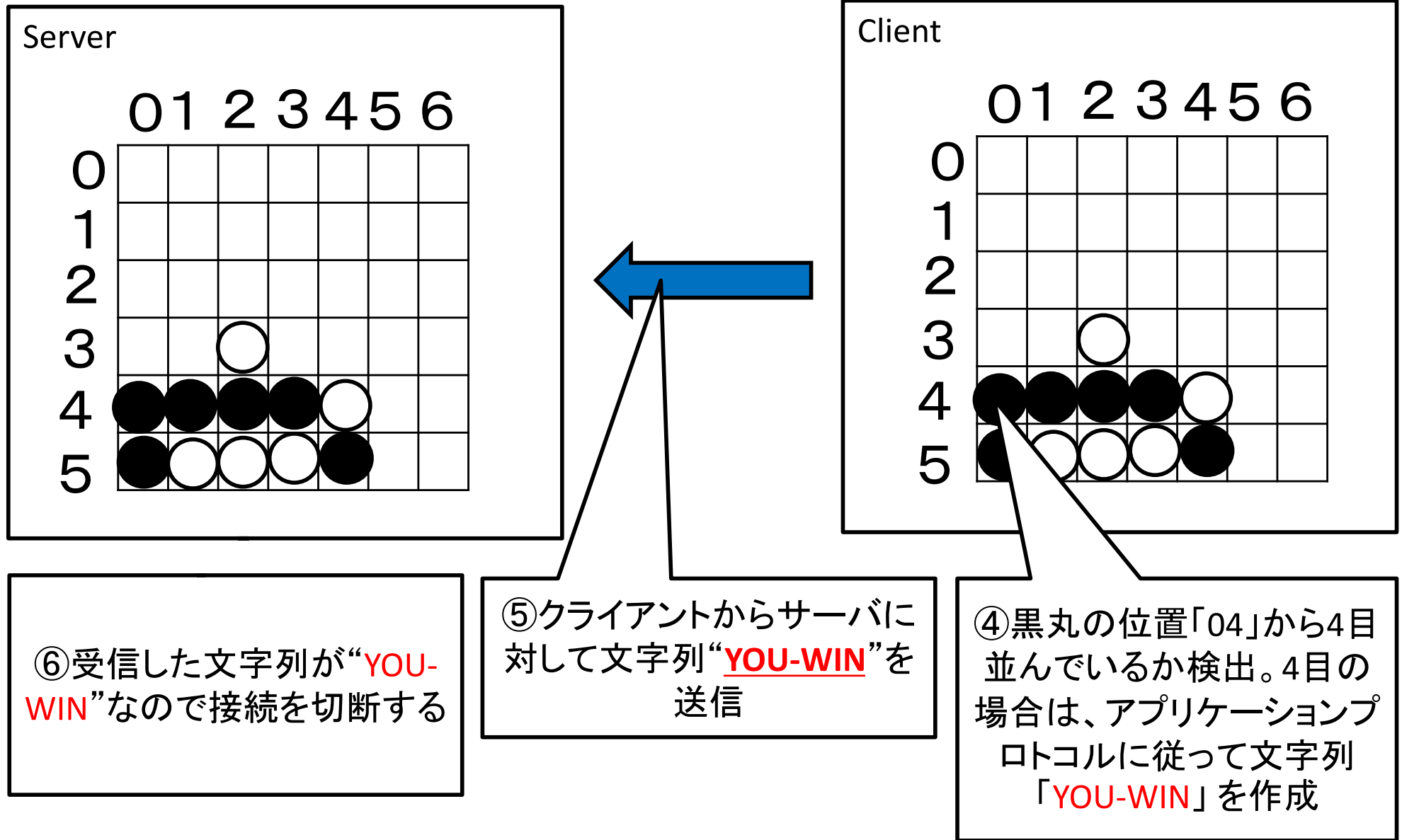
# 13. 最終テーマ

□ 4目並べとなる碁石を置いた際のプログラムの流れ(1)



# 13. 最終テーマ

□ 4目並べとなる碁石を置いた際のプログラムの流れ(2)





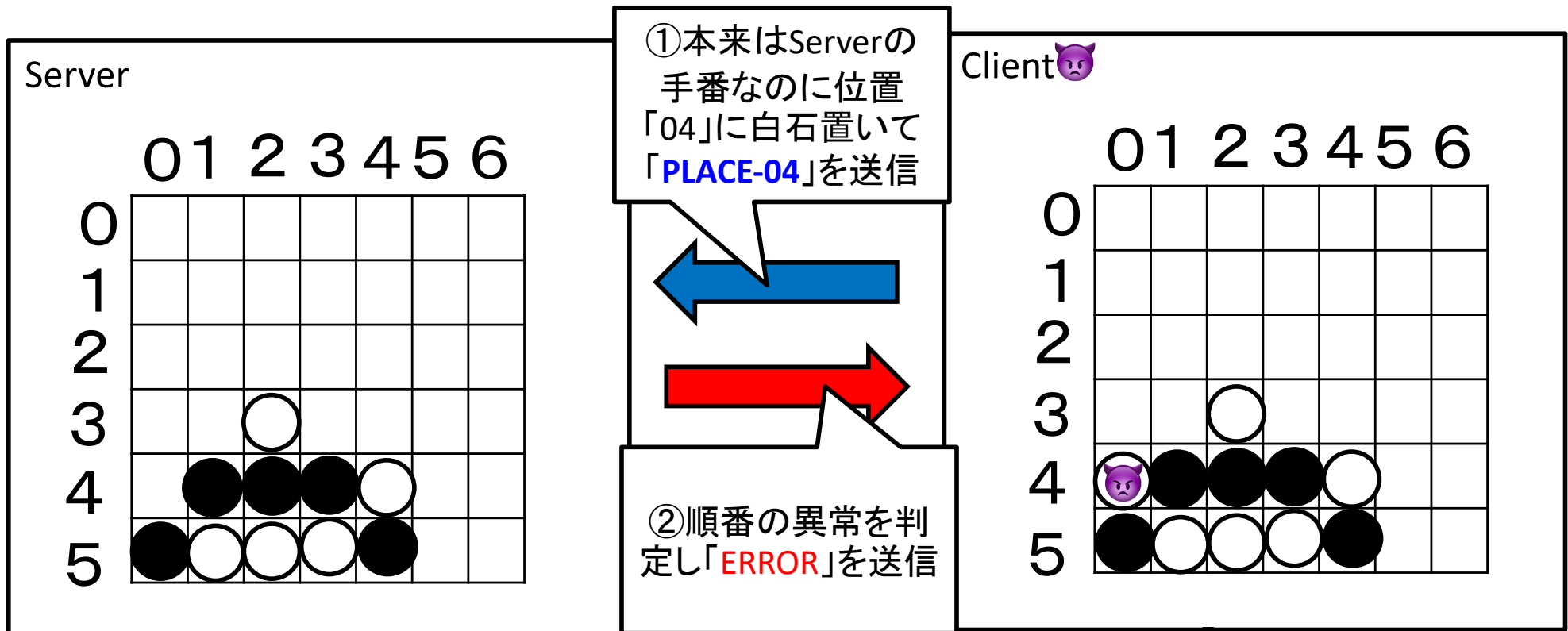
# 13. 最終テーマ

## □ アプリケーションプロトコル「ERROR」について注意

- ServerとClientが仕様を満たした**正常なプログラムの場合、Errorは引き分けの際にのみ使用・送信**される

## □ 理由：仕様により連続して石は置けないし、すでに置いてある場所には置けないようになっているから。ではなぜ必要か？

⇒相手が**チートプログラム/欠陥プログラム**の場合の対処



# 13. 最終テーマ

## □評価用サンプルプログラム

- 自ら作成したプログラム同士で対戦確認を行うと仕様を満たしていなくても正常に動作しているように見える場合があるので、**最終的には必ず評価用プログラム(実行ファイル)と対戦**させること
- 評価用プログラムの実行ファイル及び、最終テーマ(表紙)はmanabaからDL

評価用プログラムの実行ファイル: 4moku\_sample

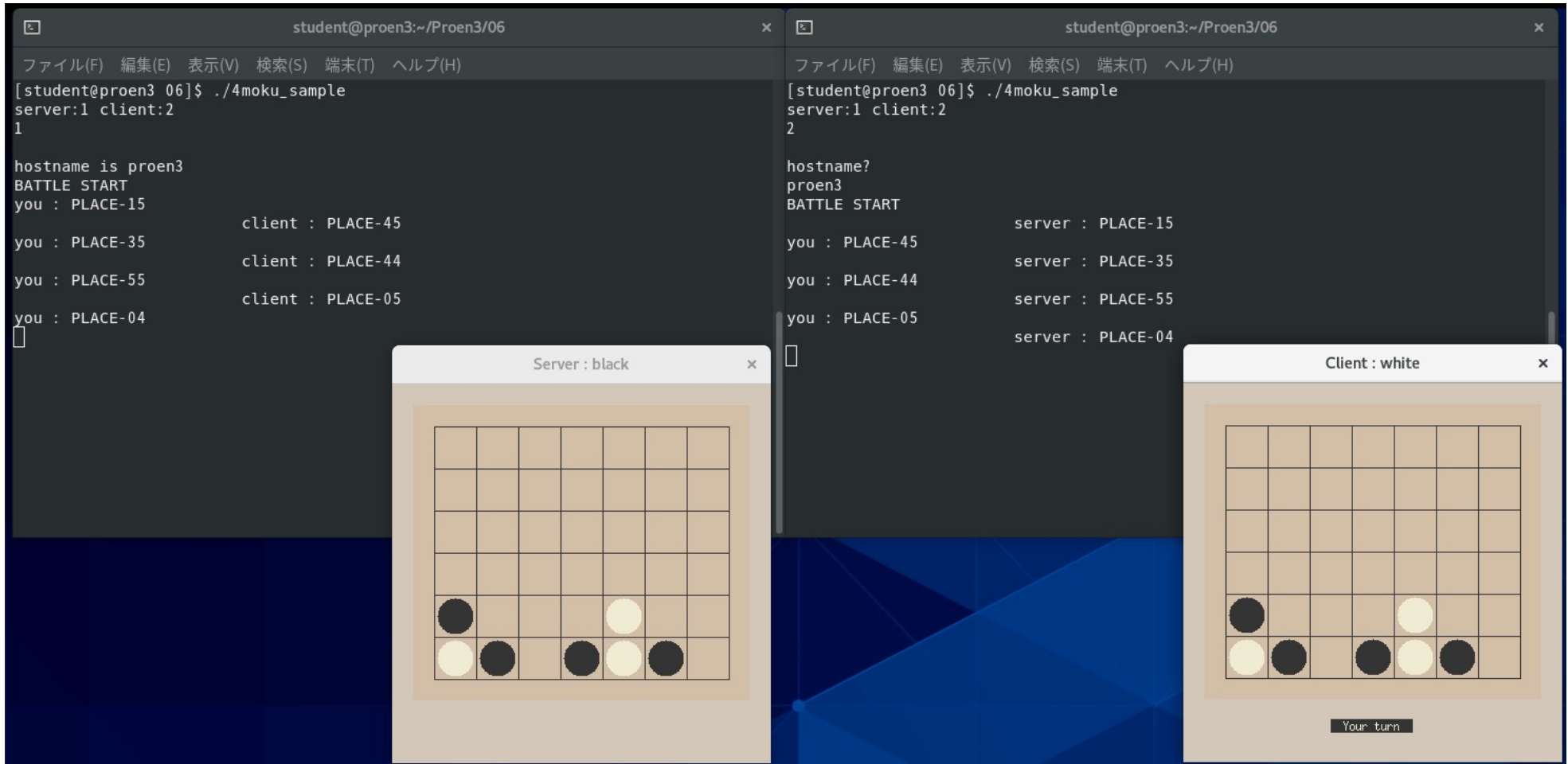
レポート表紙: (学籍番号)プロ演3最終レポート

### **【注意】**

ターミナルで評価用プログラムの実行ファイルの実行権限がないと表示された場合は「`chmod +x 4moku_sample`」を実行してから使用すること

# 13. 最終テーマ

## □評価用サンプルプログラム実行例



The screenshot displays a terminal window and two graphical game board windows. The terminal window, titled 'student@proen3:~/Proen3/06', shows the execution of a Go game program. It displays the command './4moku\_sample' and the output 'server:1 client:2'. The game board windows, titled 'Server : black' and 'Client : white', show a 10x10 grid with black and white stones placed on it. The 'Server : black' window shows a black stone at (row 8, col 1) and a white stone at (row 8, col 4). The 'Client : white' window shows a black stone at (row 8, col 1) and a white stone at (row 8, col 4). The terminal window also shows the output 'hostname is proen3' and 'BATTLE START'.

```
student@proen3:~/Proen3/06$ ./4moku_sample
server:1 client:2
1
hostname is proen3
BATTLE START
you : PLACE-15
client : PLACE-45
you : PLACE-35
client : PLACE-44
you : PLACE-55
client : PLACE-05
you : PLACE-04
□
```

Server : black

●						○			
○	●		●	○	●				

Client : white

●						○			
○	●		●	○	●				

Your turn

★サンプルプログラムは下記の点を発展させています(必須ではありません)  
カラーリング、ウィンドウ名表示、ターン(手番)表示、双方が置いた座標情報を標準出力、終了時(Error送受信後)キーボード入力でウィンドウを閉じる

# 最終テーマ(提出方法)

□提出期限: 令和3年6月14日(月)午前9時00分×(A班)  
令和3年8月2日(月)午前9時00分×(B班)

□提出物: 1) 最終テーマのプログラムソース  
2) 最終レポート(配布する表紙を用いること)  
※表紙のチェック項目及び「技術文書の書き方」を  
参考にしてレポートを作成すること

□提出方法: 下記ファイルを二つとも提出(Manabaのレポートに提出)

1) プログラムソース: 学籍番号.c

2) 最終レポート: (学籍番号)プロ演3最終レポート.pdf

※提出後、必ずアップロードされているか確認すること(重要)

# 最終テーマ

注意0: 作成プログラムはこまめにUSBメモリ等でバックアップをとること(中間課題同様)

注意1: 期限は厳守すること.

注意2: 最終課題プログラムは早めに終わらせておくこと.

注意3: プログラムを他人に渡さないこと(他班含め)  
コピーチェックを行います

注意4: きちんと実行できるプログラムを提出すること.

注意5: プログラムソースはPDFやWordで提出しないこと.

発展させたポイントがある場合はレポート表紙 & レポートに明記

# 口頭試問について(1)

- 最終レポート合格者に対して、X11とソケット通信を用いたイベント駆動型プログラミングに概念や用語の理解度を口頭試問にて確認します。
- 最終レポート合格者とは最終レポートにて、
  - 指定した全ての提出物を提出したもの
  - 指定した仕様を満たしているプログラムを提出したもの
  - 当該テーマ6回(自習日と口頭試問は含まない)のうち欠席が2回以下で中間レポートを提出・受理されているもの
- 口頭試問予定日(ManabaかTeamsにて公開):
  - 【A班】講義期間内にアップデートします
  - 【B班】講義期間内にアップデートします
  - 【再試】令和3年8月9日- 8月10日(対象者のみ)

# 口頭試問について(2)

## オンラインでの口頭試問の基本方針:

- 1コマあたり最大10名、接続時間を含めて一人10分以内
- 学生は決められた時間帯にTeamsで連絡可能状態にしておく。教員側から学生に対してビデオ通話をかけます。
- 接続不能の時は欠席と見なす。ただし通信トラブルの場合は、必ず事前にチャット等で連絡すること(別途、時間を設定します)。
- ビデオは双方On、背景はボカシOK、画像NG。レコーディング無し。指示された場合を除き、学生は資料閲覧不可。尋ねることは、基本用語・概念、レポートの内容
- 口頭試問の可否をその場で伝えます。
- 類似レポートが後日判明した場合は合格を取り消すので注意すること。

# 出席確認とアンケート回答について

講義期間内にアップデートします



# USBメモリの使いかた(重要)

課題プログラム作成中は定期的にプログラムをバックアップしておいてください

- VirtualBoxでCentOSを起動、ログイン
- USBメモリをPCに挿す
- 最初は本体OS(Windows)で認識されるので下記の手順を実行
  1. VirtualBoxの上部タブ「デバイス」→「USB」から、「General USB Flash Disk」を選択(USBによっては違う名前が表示される可能性があります。その場合は、USB接続前と後を比較し、接続後に「USB」に新たに表示された名前を選択してみてください)
  2. CentOS上でUSBメモリが認識
  3. USBメモリを取り外す際は、USBメモリ名の右横に表示されている逆三角形マーク(アンマウント)を選択

次ページから、例年よくある質問を掲載します

**【注意】**

よくある質問の回答については講義の理解度によっては勘違いしてしまう可能性もありますので、不安がある場合は必ずTeamsのチャット機能等を通じて個別にご連絡ください。

# よくある質問(1)

質問・コメント	回答
口頭試問が不安	講義資料を見直して基本的なことを復習しておいてください。
口頭試問で雑音が入っちゃうかも	全く問題ないです。安心して受けてください。
課題が遅れても出せますか	公平性の点からかなり特別な理由がない限り延期はできません。
最終課題に手がつけられない(1)：そもそも通信するための手順について ⇒第5回講義の非同期チャットプログラムを復習すること	サーバ・クライアント間で実際にソケット記述子を用いて通信している部分について要確認。第5回講義資料等から、「select関数」と「ソケット記述子を用いてどのように通信しているか」を復習してみてください（講義ではこの2つを用いてチャットプログラムを作成していますが、厳密にはselect関数はファイル記述子の変化を検知するための関数であり(マクロの使用含め)、通信部分はソケット記述子とwrite関数が担っています。

# よくある質問(2)

質問・コメント	回答
<p>最終課題について手がつけられない(2)：中間課題からどのように発展すればよいのか ⇒第5回講義の非同期チャットプログラム及び演習課題を復習すること</p>	<p>最終課題のポイントは「自分が基石を打つ」と「相手が基石を打つ」を検出する手順が違ふことに注意。下記はあくまで一例</p> <p>「自分がウィンドウで基石を打つ」のは中間課題同様、イベント(マウスクリック)で検知します。このイベントが発生した場合、中間課題ではイベント処理として座標位置の検出や基石の描画を行っていたかと思いますが、このイベント処理内に検出した座標位置を相手に送る(通信処理)を加えるのが一般的かと思います(あくまで一例)。一方で、相手側が基石を打った場合は、まずソケットの記述子に変化がありますのでこちらを検出したら基石の描画等の処理を行うというプログラム構造にする必要があるかと思います(select関数とチャットプログラム内のif文構造を用いて検出できそうですね)。</p>

# よくある質問(3)

質問・コメント	回答
<p>最終課題について手がつけられない(3)：ソケット記述子を用いて何を通信すればよいのかわからない。</p>	<p>本課題ではソケット記述子で送信すべきメッセージは「アプリケーションプロトコル」により下記の3つのメッセージに限定されています。</p> <ul style="list-style-type: none"><li>▪ PLACE-XY</li><li>▪ ERROR</li><li>▪ YOU-WIN</li></ul> <p>これ以外の文字は送ってはいけませんし、もし送られてきたとしても何も反応する必要はありません。</p> <p>上記のメッセージを作成、または判別するためにはC言語における“文字”の取り扱いが必要になります。C言語における“文字”の取り扱いについてはC言語の教科書やWebを参考にしてください。</p>

# よくある質問(4)

質問・コメント	回答
<p>ERRORメッセージ送信についてサンプルプログラムが仕様を満たしていないのでは？</p>	<p>サンプルプログラムは仕様を満たしています。下記誤解がないよう確認してください。</p> <p>サンプルプログラムや正常に自作されたプログラムでは、<u>ルールを無視したような基石は打てないようプログラムされている</u>はずですのでERRORは引き分けの際にしか送られてきません。一方で、オンライン対戦では相手側が、ルールを無視したような基石（PLACE-XY）をおけるような欠陥プログラムの可能性もあります。その際には、作成プログラム内で送られてきたPLACE-XY判断し、ルールに準拠していない場合はERRORを送信してあげる必要がありますので、そのためのエラー判断用プログラムも作成しておいてください（何度も言うようですがサンプルプログラムと対戦時はこのエラー判断用プログラム構文は実行されません）。</p>

ここまでのよくある質問の回答については講義の理解度によっては勘違いしてしまう可能性もありますので、不安がある場合は必ずTeamsのチャット機能等を通じて個別にご連絡ください。