情報実験 I
三好担当分 第2回

芝浦工業大学
システム理工学部
電子情報システム学科
三好　匠

E-mail：miyoshi@shibaura-it.ac.jp
研究室HP：http://www.minet.se.shibaura-it.ac.jp
授業支援システム：https://lms.sayo.se.shibaura-it.ac.jp/

注意

- C言語を使用します
- UNIXのソケット通信を利用します
- OSによってソケット通信の実装が異なります
- Linuxを起動して下さい
  （またはLinuxマシンにログインして下さい）
- 漢字コードはUTF-8を使用します

今日の授業

- （通信）プロトコル
  - プロトコルとは
  - なぜプロトコルが必要か
  - プロトコルと状態遷移
- 授業の流れ
  - プロトコルについて
  - プロトコルの実装例
  - プロトコルの実装課題

プロトコル

- プロトコル（**Protocol**）
  - 通信規約，あるいは通信手順
  - 通信の「仕方」に関する取り決め
- なぜプロトコルが必要なのか
  - 人間同士の意思疎通に置き換えてみると・・・
    - 英語を使ってアメリカ人と会話
    - 「いらっしゃい」「ご注文は？」「チャーハン下さい」
    - 「芝浦工大の三好ですが，○○さんいらっしゃいますか？」

約束ごと

- プロトコルが守られないと・・・
  - 電話をかけてきておいて名前も名乗らないなんて
  - レストランなら「いらっしゃい」ぐらい言え
  - 日本語がわからないアメリカ人に日本語で話かけてもねぇ
- 人間だから許容範囲？
  - 怒っている → ある程度理解できている！
  - 曖昧な会話であってもある程度成立
- もしコンピュータならどうなる？
  - 理解不可能！

既にプロトコルを考慮している!?

- ソケット通信の手順を思い出してみよう
  - サーバのソケット作成手順
    - `socket()` → `bind()` → `listen()` → `accept()`
  - クライアントのソケット作成手順
    - `socket()` → `connect()`

→ これこそがプロトコルである

通信プロトコルの階層化

- **OSI参照モデル**
  （OSI＝Open System Interconnection）

アプリケーション層	実際のサービス
プレゼンテーション層	データの表現形式
セッション層	仮想的な経路の確立や開放
トランスポート層	伝送確認，再送制御
ネットワーク層	通信経路の選択やアドレス管理
データリンク層	ケーブル上の通信方式
物理層	ケーブル，無線

OSI参照モデルとインターネット

- インターネットは**5層構造**
  - アプリケーションがセッション層以上を担当

アプリケーション	アプリケーション層
	プレゼンテーション層
	セッション層
TCP，UDP	トランスポート層
IP	ネットワーク層
Ethernetなど	データリンク層
Ethernetケーブルなど	物理層

ソケット通信を使用すると

- トランスポート層以下の通信手順を指定済
  - AF_INET・・・IP利用
  - SOCK_STREAM・・・TCP利用
- ということは・・・
  - 残すはアプリケーション層のプロトコル作成のみ
  - どういう手順でデータのやりとりを行うかを規定

アプリケーション
TCP，UDP
IP
Ethernetなど
Ethernetケーブルなど

先週の課題（再考）

- **次の要件を満たすTCP/IP通信型大文字・小文字変換プログラム（サーバ・クライアント両方）を作成せよ**
  - ◆サーバ
    - ●クライアントから送信された文字列に対し，大文字は小文字に，小文字は大文字に変換して返す
    - ●それ以外の文字は変換しない
  - ◆クライアント
    - ●キーボードから文字列を入力
    - ●入力された文字列をサーバに送信
    - ●サーバから送信された文字列を画面に出力

サーバの通信手順

- **サーバ**
  - ◆クライアントから送信された文字列を読み込む
  - ◆読み込んだ文字列に対して処理を行う
  - ◆処理結果をクライアントに送信する
- **プログラム表現（例）**

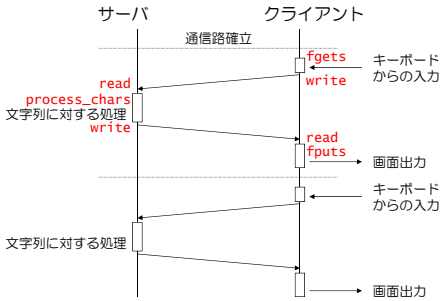
```
read( fd, buf, 1024 );
process_chars( buf );
write( fd, buf, strlen(buf)+1 );
```


クライアントの通信手順

- **クライアント**
  - ◆キーボードから文字列を読み込む
  - ◆読み込んだ文字列をサーバに送信する
  - ◆サーバから送信された文字列を受信する
  - ◆画面に表示する
- **プログラム表現（例）**

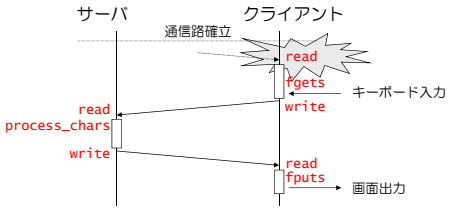
```
fgets( buf, 1024, stdin );
write( fd, buf, strlen(buf)+1 );
read( fd, buf, 1024 );
fputs( buf, stdout );
```


時系列を考慮した手順の記述



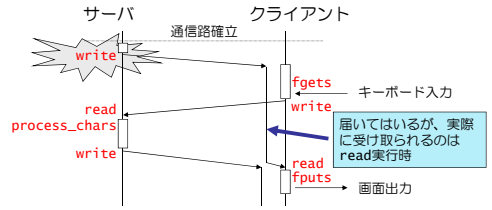
プロトコルが守られない場合（1）

- **届くはずのないメッセージをreadしてしまう**
  - → 処理は永久に進まない！！



プロトコルが守られない場合（2）

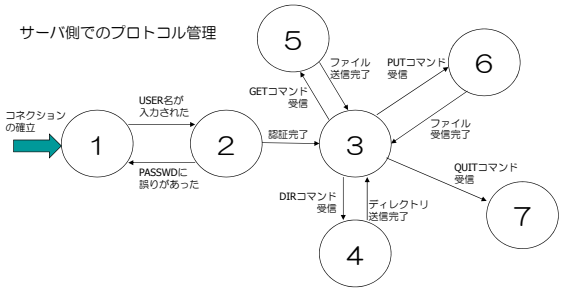
- **受け取ってくれないメッセージをwriteしてしまう**
  - → 処理がずれる！！



複雑なプロトコル管理

- **文字列の送受信のみなら非常に簡単**
- **実際のプログラムでは複雑なプロトコル**
  - ◆プロトコルの実例（FTP）
    - ●ユーザ名の送信
    - ●パスワードの送信
    - ●ディレクトリツリーの確認（ls/dir）
    - ●ファイルの受信（get）
    - ●ファイルの送信（put）
    - ●通信終了（quit）

状態遷移図を用いたプロトコル管理例

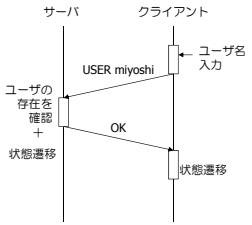


状態管理の意味

- **サーバ**
  - ◆状態を記憶しておくことで誤ったコマンドの入力を避ける
  - ◆正しい手順を踏ませる（認証など）
  - ◆許可されたユーザのみに情報を提供する
- **クライアント**
  - ◆正しい手順を踏ませる
  - ◆ユーザ・インタフェースによる入力支援を行う
  - ◆使いやすさを向上させる

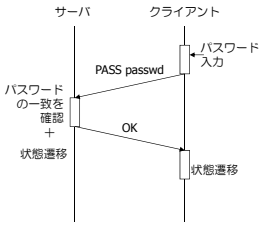
各状態での通信機能（1）

- 状態1： ユーザIDの入力
  - C→S：“USER miyoshi” をメッセージとして送信
  - S： メッセージの受信
    - 第1引数・・・コマンド名
    - 第2引数・・・ユーザID
  - S→C：“OK” をメッセージとして送信
    状態の遷移
  - C： メッセージの受信
    - “OK”なら状態が遷移
    - “NG”なら状態遷移なし



各状態での通信機能（2）

- 状態2： パスワードの入力
  - C→S：“PASS passwd” をメッセージとして送信
  - S： メッセージの受信
    - 第1引数・・・コマンド名
    - 第2引数・・・パスワード
  - S→C：“OK” をメッセージとして送信
    状態の遷移
  - C： メッセージの受信
    - “OK”なら状態が遷移
    - “NG”なら状態遷移なし



課題

- 次の要件と次ページ以降の通信規定を満たすTCP/IP通信型クイズ出題クライアントプログラムを作成せよ
  - サーバ（作成不要）
    - 認証されたユーザに対しクイズを出題，5問正解したら秘密のメッセージを送信する
    - すべての状態において，クライアントからの要求に対して受動的に動作を行う
  - クライアント
    - クイズサーバに対応したユーザインタフェースを実現する

通信規定（1/7）

- 状態の規定
  - サーバには以下の状態が存在
    - ユーザ名を受け付ける状態（状態０）
    - パスワードを受け付ける状態（状態１）
    - クイズ出題を受け付ける状態（状態２）
    - 解答を受け付ける状態（状態３）
    - 秘密のメッセージの送信を受け付ける状態（状態４）
- クライアントでは，サーバがどの状態にあるかを判断して，必要なデータ送信を実施すること

通信規定（2/7）

- ユーザ名受付状態（状態０）
  - ユーザ名として，自分の学籍番号を使用すること
    - 例： bp15150（アルファベットは小文字）
  - ユーザ名の送信には，「USER ＜ユーザ名＞」の形式を使用すること
  - 正しいユーザ名が入力された場合，サーバは文字列“OK”を返し，パスワード受付状態（状態１）に遷移すること
  - 誤ったユーザ名が入力された場合，サーバは文字列“NG”を返し，状態を遷移しないこと
  - 「USER」以外のコマンドが入力された場合，サーバは文字列“ERROR”を返し，状態を遷移しないこと

通信規定（3/7）

- パスワード受付状態（状態１）
  - パスワードとして，自分の学籍番号（CD付）を反対から書いた文字列を使用すること
    - 例： 05151pb
  - パスワードの送信には，「PASS ＜パスワード＞」の形式を使用すること
  - 正しいパスワードが入力された場合，サーバは文字列“OK”を返し，クイズ出題受付状態（状態２）に遷移すること
  - 誤ったパスワードが入力された場合，サーバは文字列“NG”を返し，ユーザ名受付状態（状態０）に遷移すること
  - 「PASS」以外のコマンドが入力された場合，サーバは文字列“ERROR”を返し，状態を遷移しないこと

通信規定（4/7）

- クイズ出題受付状態（状態２）
  - クイズの出題要求には，「QUIZ ＜現在の正解数＞」の形式を使用すること（すなわち，正解数をクライアントで保持しておく必要がある）
  - 「現在の正解数」が正しく入力された場合，サーバはクイズを出題（すなわち，クイズの文章を送信）し，解答受付状態（状態３）に遷移すること
  - 誤った正解数が入力された場合，サーバは文字列“NG”を返し，状態を遷移しないこと
  - 「QUIZ」以外のコマンドが入力された場合，サーバは文字列“ERROR”を返し，状態を遷移しないこと

通信規定（5/7）

- 解答受付状態（状態３）
  - 解答の送信には，「ANSR ＜答え＞」の形式を使用すること
  - 正しい解答が入力された場合，サーバは文字列“OK”を返し，正解数を１増加させること
  - 間違えた解答が入力された場合，サーバは文字列“NG”を返すこと
  - 正解数が5になった場合，秘密メッセージ受付状態（状態４）に遷移すること，それ以外の場合にはクイズ出題受付状態（状態２）に遷移すること
  - 「ANSR」以外のコマンドが入力された場合，サーバは文字列“ERROR”を返し，状態を遷移しないこと

通信規定（6/7）

- 秘密メッセージ受付状態（状態４）
  - 秘密のメッセージは「GET MESSAGE」にて要求すること
  - 正しいコマンドが送られてきた場合，サーバは秘密のメッセージ文章を返し，状態を遷移しないこと．
  - 引数部に誤りがある場合（すなわち「MESSAGE」以外の文字列を受信した場合），サーバは文字列“NG”を返し，状態を遷移しないこと
  - 「GET」以外のコマンドが入力された場合，サーバは文字列“ERROR”を返し，状態を遷移しないこと

通信規定（7/7）

- 終了メッセージ
  - サーバは「QUIT」を受信すると“GOOD BYE”を返し，どの状態にいても接続を切断すること
  - クライアントは「GOOD BYE」を受信すると，接続を切断すること
- その他の注意
  - write()を実行する際には，送信する文字列長（NULL文字を含む）に合わせて第3引数を設定すること
  - サーバは「STAT」を受信すると状態情報を返すようになっているので活用すること
  - サーバのサンプルプログラムをscombよりダウンロード可能

クライアントの条件

- クライアントでは通信規定に基づき的確なユーザインタフェースを実現すること
  - 「ユーザ名を入力して下さい→」の表示
  - サーバからのメッセージを受信したときには，それをそのまま表示するのではなく適当な文章に変換して表示
  - NG → 「パスワードが違います」など
- 必要な状態を記憶しておくこと
  - クイズ正解数は利用者が覚えておくのではなく，クライアントソフトで自動的に記憶可能である
  - クライアントによる支援

提出方法

- 新授業支援システム「Scomb」を使用
  - クライアントプログラム，及び実行結果（次スライド参照）
  - **提出期限：7月5日 13時10分**
  - ユーザインタフェースの実装が課題ですので，解答をキーボードから入力するようにプログラムを作成すること
- プログラムの実行による採点
  - 次週の情報実験Ⅰの授業開始直後にクイズ大会を実施します
  - 速く解けた方にプレゼントを差し上げます！
  - 提出したプログラムとは別に，高速解法プログラムを開発しても構いません！

注：問題は入れ替わる可能性があります
5問間違えるとクイズ出題がとまります

実行結果取得方法

- 以下のクイズサーバに接続して実行結果をとること
- クイズサーバの情報
  - IPアドレス
    - 172.29.144.26（iexp100.minet.se.shibaura-it.ac.jp）
    - 172.29.144.27（iexp101.minet.se.shibaura-it.ac.jp）
      （どちらも同じ，学外からはアクセスできないので注意）
  - ポート番号
    - 34401 または 34402
- 動作確認用のプチ・クイズサーバを配布予定
  - 授業支援システムScombからダウンロード可能

参考：便利な関数

- strcat(char *s1, char *s2)
  - 文字列s1の後ろに文字列s2の中身をくっつける関数
  - 実行例： strcat(s1, s2);
    - s1に “abc” が，s2に “def” が入っている場合，関数実行後，s1には “abcdef” が入る
- sprintf(char *s, "書式指定子", 変数の列)
  - printf や fprintf と同じ感覚で，指定した文字や変数の値を文字列sに書き込む
  - 実行例： sprintf(s2, "USER %s", s1);
    - s1に “bp13150” が入っている場合，関数実行後，s2には “USER bp13150” が入る