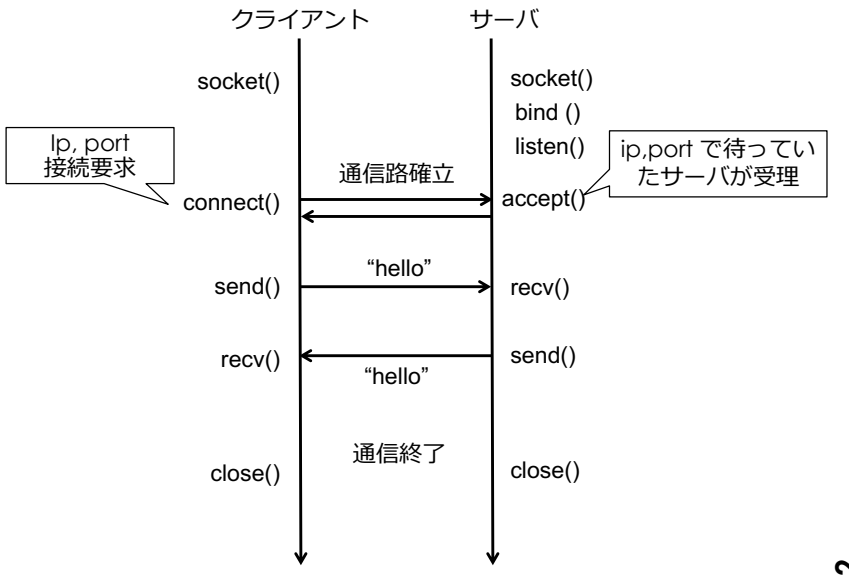
システムプログラミング
ソケットプログラミング 2
(server)

芝浦工業大学 情報工学科
菅谷みどり

TCP/IP通信路: C/S システムコール



Echo クライアント/サーバ

- **Echo クライアントプログラムの動作**
  - コマンドラインから2つの引数(IP, port)
  - 指定されたIP/ポートで動作しているサーバに接続
  - キーボードから1行単位で入力 > サーバに送付
    - サーバは同じ文字列を送る
  - 送られてきた文字列を受け取り、結果を画面に表示

- **Echo サーバ**
  - コマンドラインから2つの引数(IP, port)
  - Socket を作成して、指定されたポートに bind
  - Port でクライアントからの要求待ち(listen)
  - 接続要求(connect) があった場合受理(accept) する
    - 通信路の確立
  - データを受信し、画面に表示

課題1

- **Echo サーバプログラムを作成しよう**

動かしてみよう

- **1. サーバを起動する**
  - 一つサーバを決めて、そこでサーバプログラムを起動する
    - $ifconfig コマンドで自分のマシンのIP アドレスを調べる
    - 調べた IP アドレスと、Well-known ポート以外のポートを引数に指定する

```
$ ./server [現在のPCのIPアドレス] 12345
```

  - IP アドレスの例 172.16.24.17
  - ➔ サーバは while でポートからの入力を待つ(listen) 状態に入る

- **2. クライアントからサーバにアクセスする**
  - サーバが起動している IP & ポートにアクセスする

```
$ ./client [1. のサーバのIPアドレス] 12345
```


Ifconfig によるIP の確認

```
$ ifconfig
eth0      リンク方法:イーサネット  ハードウェアアドレス 00:0C:29:BB:92:BC
          inetアドレス:172.16.24.13 ブロードキャスト:172.16.24.255 マスク:255.255.255.0
          UP BROADCAST RUNNING MU
          RXパケット:55081878 エラー:0
          TXパケット:35669040 エラー:0
          衝突(Collisions):0 TXキュー長:1000
          RX bytes:73726745597 (70311.3 Mb)  TX bytes:12149433274 (11586.6 Mb)

lo        リンク方法:ローカルループバック
          inetアドレス:127.0.0.1 マスク:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RXパケット:6709 エラー:0 損失:0 オーバラン:0 フレーム:0
          TXパケット:6709 エラー:0 損失:0 オーバラン:0 キャリア:0
          衝突(Collisions):0 TXキュー長:0
          RX bytes:513785 (501.7 Kb)  TX bytes:513785 (501.7 Kb)
```

ifconfig を実行しているマシンのeth0 の IP アドレス

実行例

サーバプログラムが起動するホストの IP アドレス

Well-known 以外の空いているポート

```
$ ./server 172.16.24.13 12345
connect from 172.16.24.12: 48918
<== aaa

<== bbb

<== hello

<== exit
```

```
./client 172.16.24.13 12345
==> aaa
<== aaa
==> bbb
<== bbb
==> hello
<== hello
==> exit
<== exit
```

サーバプログラムが起動している接続先ホストの IP アドレス

サーバが監視しているポート

課題2

- **Server.c**
  - １つのクライアントから接続されると，そのクライアントにかかりきりになって，他のクライアントにはサービスを提供できないという問題がある．
- **2-1）**
  - 複数のクライアントを実行する端末を立ち上げて，確認しよう
    - /client 172.16.24.13 12345
    - /client 172.16.24.13 12345
- **2-2）**
  - 前回の資料（ソケット１）でアクセスした標準のecho サーバ（ポート７で動作）は，複数のクライアントを受け付けることができるかを確認しよう
    - ./client 172.21.39.17 7
    - ./client 172.21.39.17 7

複数のクライアントを同時に扱う方法

- **複数のクライアントに対してサービスを同時に提供するには，次の方法がある**
  - クライアントごとに fork() でプロセスをコピーする
  - クライアントごとにスレッドを生成する
  - select() を用いて，複数の接続を監視し，データが届いたことを確認してから読み込む

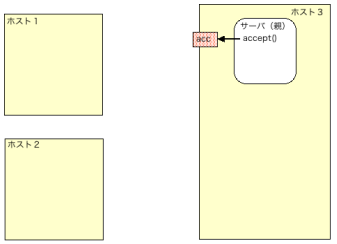
解説

- **サーバが accept で接続要求待ち**

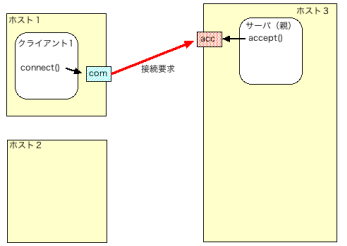
※ 図中

acc ➔ sockfd

com ➔ new_sockfd

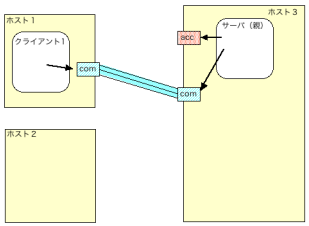


- **サーバが accept 中にクライアントが connect**

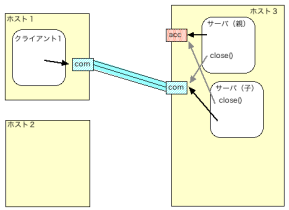


解説

- **Accept の結果，通信用ポート (new_sockfd) が作られる**

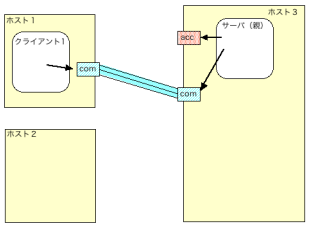


- **Fork() して親子に分かれる**

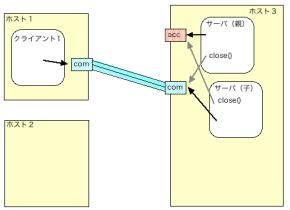


解説

- **Accept の結果，通信用ポート (new_sockfd) が作られる**

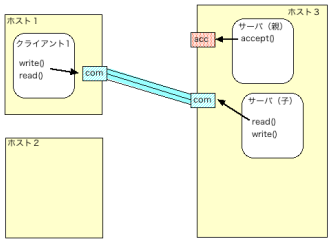


- **Fork() して親子に分かれる**

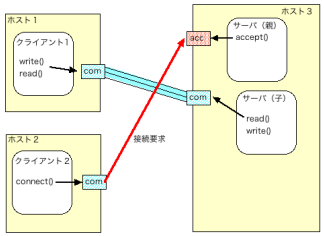


解説

- 親は子用の通信ポート(new_sockfd)を，子は親が作成した通信ポートを閉じる(close())

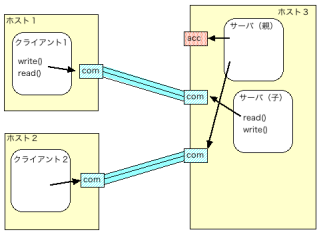


- 親は再び accept() で待ち，
- 子プロセスはクライアントに対して Read/write する

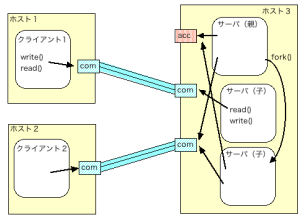


解説

- サーバがまた別のクライアントから接続要求を受け付ける

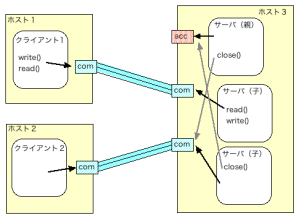


- Accept() の結果，もう一つ
- 新たに通信用ポートが作られる
- 同じ(

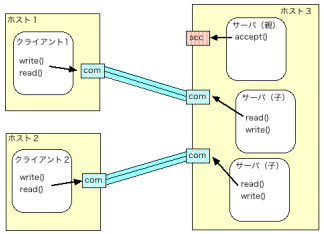


解説

- Fork() して親子に分かれる



- 親は通信用ポートを closeし，子は接続用ポートを close する



最終課題

- **課題１）複数クライアント処理サーバ**
- **課題２）サーバクライアントの仕組みを使って，サービスアプリケーションを考えて作成する**
  - 例） サークライアントの仕組みを使ったネットワークゲーム
  - サーバを介したデータ共有の工夫ができることが望ましい
    - 共有メモリの利用など
  - 条件
    - 協力者がいる場合，2人までの取り組みはOKとする
    - サーバ側には、マルチプロセス、マルチスレッドなどの技術を組み込んで、複数のクライアントに対してサービスできるようにする
- **ソースコード & レポート（動作内容の説明）の提出**
  - 演習２）については，特に，作成したサービスについて
  - １）目的
  - ２）利用方法
  - ３）工夫した点
  - ４）ソースコードの解説
  - を資料に記載、レポート＋コード提出
  - 締切は授業中に示す