ブラウザのフタを開けてHTTP体験しよう

深町賢一 (公立千歳科学技術大学)

ふだん使っている道具は便利だけれど
なんかモヤモヤしませんか？

モヤモヤの理由

- ふだん使っている道具、たとえばWWWブラウザが、どう動いているのか？
  裏側が気になったことはありませんか？ (-> モヤモヤする)
- フレームワークとかSDKとか便利なものを使えば、さくっと色々なアプリを作れはするのですが、 いったい何がどう動いているのか分からない（-> モヤモヤする）
- モヤモヤの理由、それは**ブラックボックスすぎるから**だと思うのです

フタを開けてみよう

- モヤモヤするならフタを開けて裏側をのぞいてみましょう。 案外かんたんな理屈です
- フタを開けると、
  - 動作原理（理屈）が分かります
  - 原理が分かれば、エラーメッセージから原因が類推できるようになります
  - ほかの技術も似たり寄ったりなので、新技術の目利きも出来るようになります
  - そして、なにより楽しいではないですか、フタを開けるのって
- 10年、20年先のキャリアにつながることがあります
  (楽しいことで御飯が食べられたら最高ですよね?:-)

(脚注) あんがい、どれも理屈は簡単なのですけど、 それを10万台のサーバで連携して24時間運用しろ！というのは大変なのです。
そこが御飯を食べられる所以でもあり、面白いところでもあります

あなたは10年後なにをしていたいですか?

- どうせなら好きなことで食べていきたいよね?
- 就職はゴールではありません。けれども、 できることなら長く技術の仕事をして食べたいよね?
  - 10年後、20年後、... 50年後
- キャリアの例
  1. 特定の(せまい)分野を極める
  2. プロジェクトを技術監修する **(上から下まで、ひととおり分かる) ITアーキテクト**などと呼ばれる立場
     - 現実には、非常に得意な分野と、そうでない分野の濃淡があるでしょう。 真のオールラウンダーなど滅多にいません
     - 得意分野として**ITインフラ**はいかが？
- ではHTTPの裏側の体験へ進みましょう

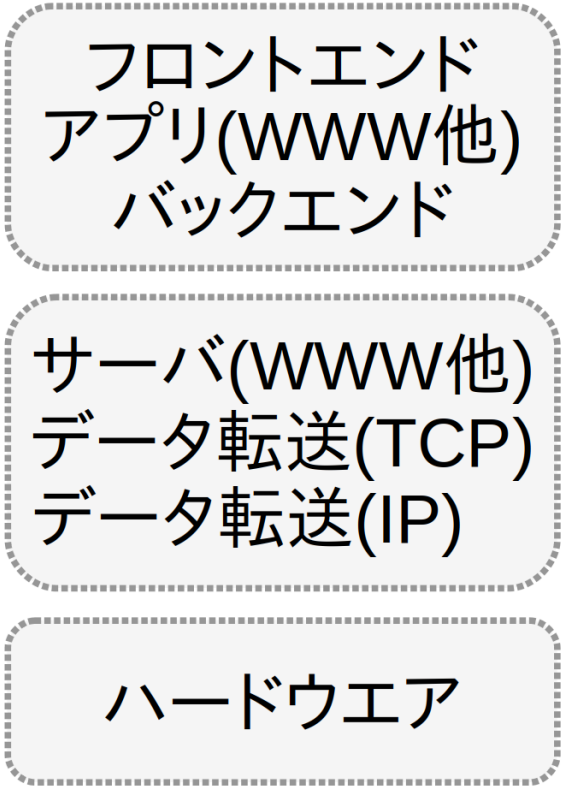


図: かなり大味のイメージ

(脚注1) いつも授業はココから (脚注2) 異論は、あるでしょう:-) なお、肩書きは業界や会社によって異なるので気をつけてください

本動画について

(脚注) 脚注にコメントが書いてありますが、大半は飛ばしていくので、なにかあれば質問コーナーでどうぞ

本動画について

- ITインフラストラクチャ(以下ITインフラ)の演習です
  - ふだんから**実務前提のITインフラの演習授業**群を行っています
  - これらの演習授業群の一部を切り出してきたデモ（演習体験）です
- 想定されている受講者（視聴者も含）
  - フタがあったら開けてみたい！
  - ~~手短に使い方をおしえてください~~

(脚注) 2頁先の自己紹介のとおり、 インターネットサービスプロバイダ(ISP)にいたので、いちおうインフラエンジニア枠なのです

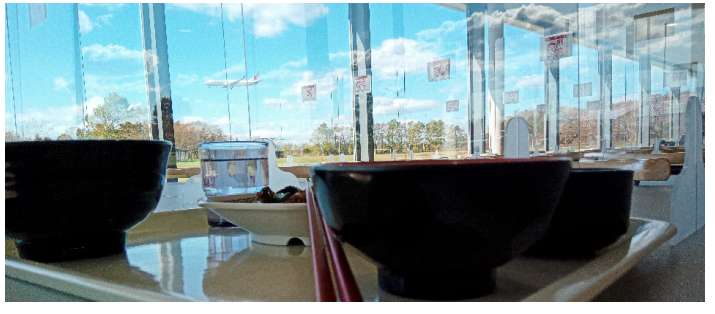
本動画の構成

1. はじめに (イントロ、本動画について等,5分程度, Aパート) ＊今ここの途中
2. ハンズオン (30-35分程度, Bパート)
    - **動作原理**がよくわかる例題を一緒にいくつかやってみましょう
3. おわりに (キャリア的な話 後編,5分強, Cパート)
4. 質問コーナー
5. 付録 (紹介のみ, スライドの最後の10枚ほど)

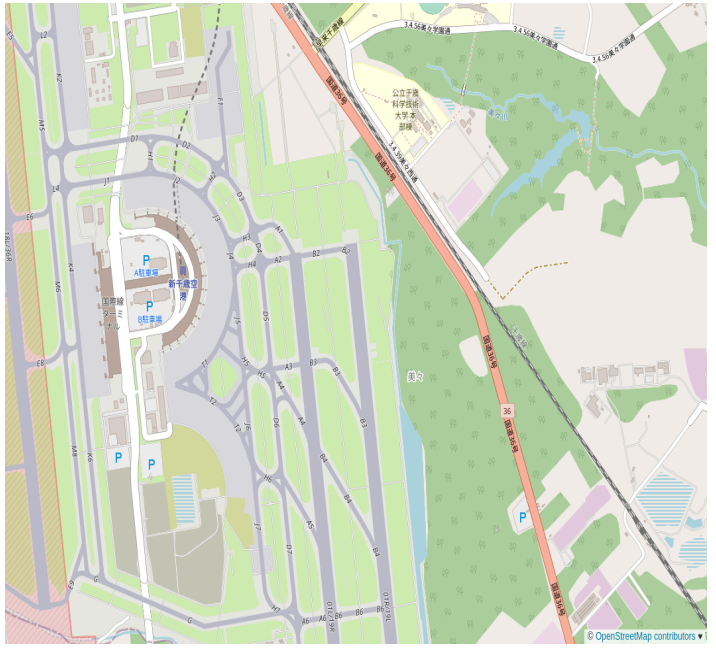
自己紹介: このあたりからやってきました

[経歴]

- フリーソフトウエア, >30年, >20万行? github
- 本来の専門は理論物理学ですが、インターネットの方が楽しい(業界あるあるですね?)
- 東京工業大学 -> IIJ(日本初ISP) -> (私立)千歳科学技術大学 -> 公立千歳科学技術大学
- 環境: NetBSD, Debian, Chromebook



写真(上) 学食からの風景: ほぼ滑走路端のレストラン。 写真(下) 左端が大学、工場(EPSON)の上に建設中の半導体工場(Rapidus)工事のクレーンが見えてます



地図(中央上あたりが大学,左が新千歳空港,右中でRapidusの半導体工場建設中)

[余談]推しのアニメ製作会社は富山県のP.A.Worksです:-) ; 図(右)(C) OpenStreeMap, available under the Open Database License.

ハンズオン事前講習会

- 環境構築/確認 -

(脚注1) 事前に講習会を行っています。 自習や研修等で本教材を利用する方は、この部分を各自で行ってください

動画収録の本番では、この事前講習会の部分を軽く確認だけしつつ、スライドを飛ばしていきます

演習環境の準備と確認

- 受講者の大前提 (***注意***: ここを事前講習会で確認します)
  - PC: 安定したインターネット接続環境のあるPCを用意できる
  - OS: sshコマンドだけは利用できる
- 演習環境にsshでリモートログインして演習を行います
  - **この動画専用の演習環境**を用意しました
  - 事前にユーザ名とパスワード、サーバ名を配布します。 この情報で演習環境にsshでログインしてください。 うまくいかない場合は、 別資料「Windowsでsshコマンドが使えるか？その確認と対処方法」 (PDF) (slides) を参照してください。それでもダメな場合は事前講習会(相談会)へ参加を
  - ターミナルを2つ開き、それぞれでsshログインしてください。サーバ用と動作体験用です

[実行例] 割り当てられたユーザ名が **user99** の場合に実行する**ssh**コマンド（**Windows Terminal**）

```
PS \Users\Your-Username> ssh user99@www.user99.demo.fml.org
```

(脚注1) このページ以降は、演習環境にログインできている想定となります

基本用語(事前講習)

- Unixオペレーティングシステム (発音：ゆにっくす)
  - 世間では「黒い画面にコマンドを打ちこむ謎の黒魔術」的な何かと思われていそうです;-)
  - このテキストではDebian GNU/Linuxのことです
    (初心者は50年以上にわたるUNIX開発の複雑な系統図なぞ気にしなくてOK)
- IPアドレス (発音：あいぴーあどれす)
  - コンピュータを識別するための数字で172.18.0.4のように10進数4つの組で表記したもの
  - 本演習では意識しなくても作業できます。ただし演習の中でIPアドレスが表示されることがあります
- ドメイン名
  - ドット(.)でつないだ文字列で組織名やサーバ名などに使われるもの
  - サーバ名のwww.user99.demo.fml.orgやURLのhttp://www.user99.demo.fml.org/

(脚注) よく分からなくても演習には差しつかえありません

基本用語(事前講習)

- WWW (World Wide Web略してW3やWeb, 1989-)
  - 発音は「わーるどわいどうえぶ」もしくは「だぶりゅーすりー」らしい、あとは略して「うえぶ」
  - 一般人がイメージするインターネットのこと(1993/09以降);
    （インフラエンジニア目線では）空前のヒットを飛ばしたアプリケーションのひとつ
  - WWW = コンテンツ(HTMLなど) + 転送システム(HTTP)
- HTML (発音：えいちてぃーえむえる)
  - `index.html`などのハイパーテキストというコンテンツを記述する言語
  - 適当に書いてもブラウザが何とかしてくれる(ので動作確認なら正確さは気にしなくてOK:-)
- HTTP (発音：えいちてぃーてぃーぴー)
  - WWWのコンテンツ転送時の約束事(プロトコル)の名称

[参考] 1993年、WWWブラウザ NCSA Mosaicは世界に色を灯したのです

登場するコマンドたち(事前講習)

- コマンド
  - コンピュータに与える命令で擬似英語っぽい単語もしくは文章、スペース区切りに注意
- ssh
  - リモートのコンピュータにログインするためのコマンド(発音：えすえすえいち)
- curl
  - ダウンロードやブラウザの代わりが出来る便利なコマンド(発音:しーゆーあーるえる or かーる)
  - HTTPを話すことが出来る
- sudo
  - 引数を管理者権限で実行できる特殊なコマンド(発音:えすゆーどぅー or すどう(日本人限定?))。
  - 管理権限が必要な作業の時に使う。 Linuxのシステム構築で頻出するコマンド
- python3
  - プログラミング言語Python(発音：ぱいそん)のプログラムを実行するコマンド（インタープリタ）
- telnet
  - 本来はリモートのコンピュータにログインするコマンド (発音：てるねっと)
  - 本演習ではWWWサーバへの(TCP)接続に使用

(脚注) もともとUnixにsu(えすゆー)コマンドがあって、 のちにsudo登場なので、すなおに「えすゆーどぅー」な気がするんですけどね

[凡例] Unixマニュアルorドキュメントの読み方(事前講習)

```
[コマンドの記述例]

$ sudo python3 www.py
```

- 左端の`$`をプロンプトと呼んでいます。**OSが「入力まち」を意思表示**していると考えてください
  - プロンプト`$`部分はOSやユーザごとに異なるので、あくまでも一例です
  - 通常、**左端にある特殊文字部分**は、プロンプトとして無視してください（そのうち慣れます）
- **英語なので空白区切り**です。初心者は日本語のつもりか連続して文字を打ちこみがちなので注意
- ユーザが打ちこむ部分は `sudo python3 www.py` です。そしてENTERキーを押します
  - 通常マニュアルにENTERキーは書きませんので、ユーザが頭の中で補完してください
  - 雑誌や書籍では、組版(くみはん)の際にデザイナさんがENTER印を書きこんでくれているのです
- 上の例では、sudoが実行するコマンド、sudoコマンドの引数が`python3 www.py`です

ハンズオン

おしながき

1. (演習環境にsshでログインする、2つのターミナルで二重にsshしてください)
2. ホームページを作ろう
   - [例題] WWWサーバを起動し、ホームページがブラウザで見えることを確認しよう
3. ホームページの動作の裏側を見てみよう
   - [例題] curlコマンドでHTTPをしてみよう
   - [例題] telnetコマンドで手動HTTPしてみよう
4. WWWサーバと会話しよう（HTMLで値をWWWサーバに送ろう）
   - [例題] ジャンケンWeb APIサーバとジャンケンしよう
   - [例題] ジャンケンWeb APIサーバとtelnetで(手動)ジャンケンしよう
5. 構成図を書いてみよう
6. [自由課題] (出題のみ)

(脚注) このページで気になるキーワードを一つ見つけましょう(5秒待つ)。 思いつかないなら、どこ？を気にしてみましょう

[凡例] テキストの書き方(事前講習兼もういちど)

- ユーザ名について ... sshする際のユーザと演習環境のユーザは異なります
  - (sshする際のユーザ名はuser01などとなっていて、各自ことなりますが)
    演習環境へログインすると、**ユーザはadmin、ホームディレクトリ(フォルダ)は/home/admin**です
  - (授業ではAWS Academyを用いてEC2を作成し、その上で作業しています)
    本テキストも他の授業資料とあわせるため、AWS提供のDebianのデフォルト値に合わせてあります
- **困った時はCtrl-C。**たとえば**www.pyの止め方もCtrl-C**です(書いてないけど)
  - **Ctrl-Cは「CtrlとCのキーを同時に押す」操作で、実行中のコマンドを途中で止める**時の定番
- 本動画特有の注意点
  - この動画では、実際に手を動かしている想定のところに**「(アーカイブ動画で見ている人は)一時停止して手を動かしてみましょう」**というスライドが挿入されています。 **アーカイブで見ている方は、動画を一時停止して、手を動かしてから、再生をリスタートしてください**
  - 演習時間を30分程度におさえるため、ダウンロードとコピー＆ペーストのみにしてあります。
    もう少し時間のかかるものは、付録に「自由課題」として収録してあるので、 各自で取り組んでみてください。ちなみに、自由課題の難易度は「やさしい」から「むずかしい」まで色々です

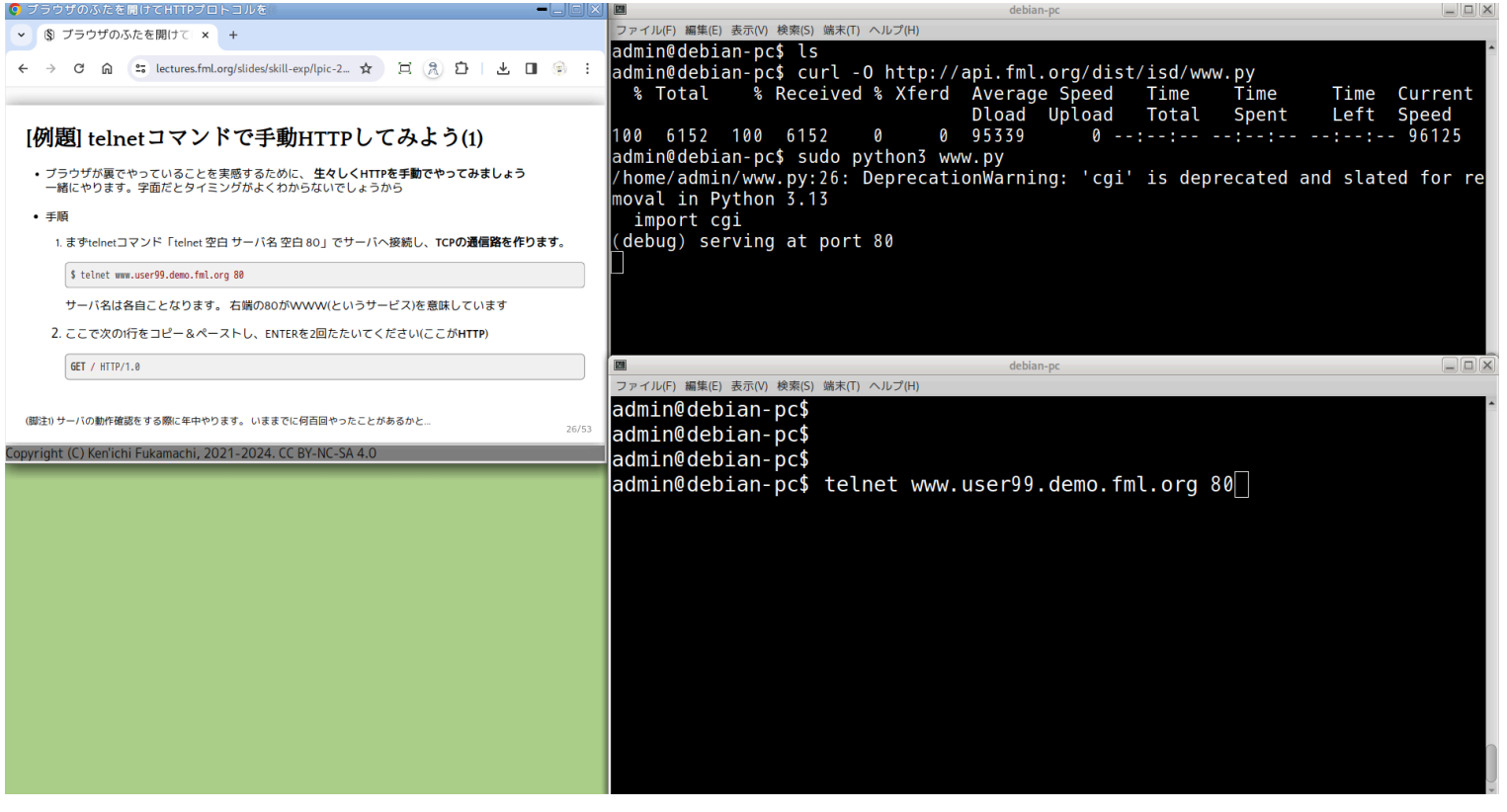
(脚注1) 自由課題は出題のみです。 いまのところ自由課題の解答編の動画を作る予定はありませんが、リクエスト数しだい？

(脚注2)AWS Academyについては付録 Eを参照

[凡例] www.pyの止め方

```
[実行例]
admin@user99:~$ sudo python3 www.py
/home/admin/www.py:26: DeprecationWarning: 'cgi' is deprecated ... (<-警告文)略 ...
(debug) serving at port 80

   ... (止めたいときは、ターミナルにCtrl-Cを打ちこむ) ...

^CTraceback (most recent call last):
  File "/home/admin/www.py", line 200, in <module>
    httpd.serve_forever()
  File "/usr/lib/python3.7/socketserver.py", line 232, in serve_forever
    ready = selector.select(poll_interval)
  File "/usr/lib/python3.7/selectors.py", line 415, in select
    fd_event_list = self._selector.poll(timeout)
KeyboardInterrupt
admin@user99:~$
```

[参考] 現在の皆さんのデスクトップは、こんな感じのはず

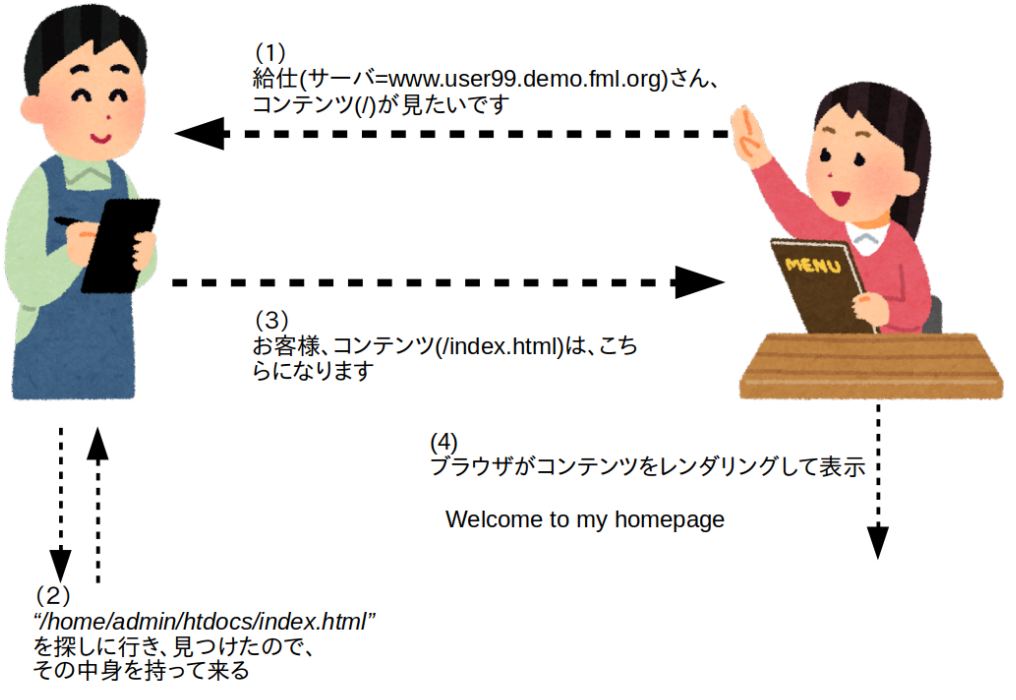


(脚注) 演習には、ターミナルが２つ、ブラウザが１つ必要です。 この画像ではブラウザ画面がZOOM相当の内容ですが、本当のデスクトップにはZOOMとブラウザが別々に出ているでしょう

[解説] ホームページ（www）のしくみ

URL(`http://www.user99.demo.fml.org/`)をブラウザでクリックすると

1. WWWサーバ `www.user99.demo.fml.org` (www.pyを動かしているサーバ)にリクエストを送ります
2. サーバ内部では、URL右端の `/` を `/index.html` と解釈して、中身(コンテンツ)を取り出し
3. サーバはブラウザにコンテンツを送り返します
4. ブラウザがコンテンツ(HTMLなど)を解釈した結果を表示します(レンダリング)

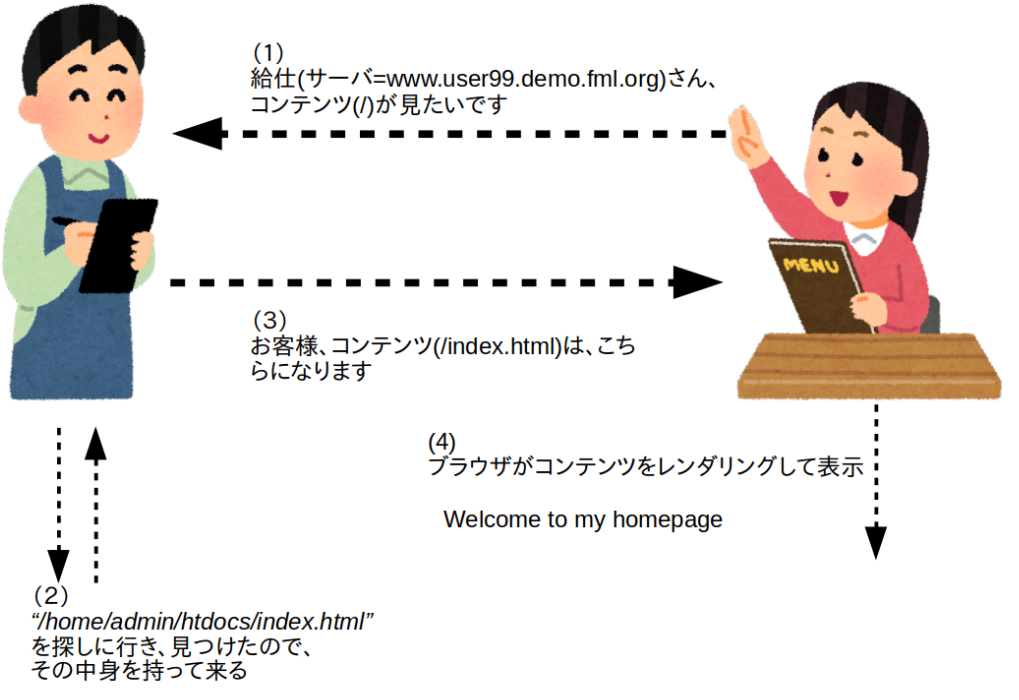


(脚注) これを「サーバ(給仕)・クライアント(お客様)モデル」と呼んでいます

[解説] URLの読み方

```
[例]
URL   http://www.user99.demo.fml.org/
URL = サーバ名＋見たいコンテンツ
```

- サーバ名が `www.user99.demo.fml.org`
- サーバ名より右側の部分(ここでは`/`)がサーバ上でのコンテンツの場所の指定(Unix用語のパス)
  - 注: `/`=`/index.html`は相対位置です。 WWWサーバ(www.py)には、あらかじめ `/index.html` の実在位置を`/home/admin/htdocs/index.html`と解釈する設定が入っています



(脚注) Unix用語では/home/admin/htdocs/index.htmlを絶対パス(Absolute Path)と呼びます。反対語が相対パス(Relative Path)

[例題] ホームページを作ろう(1)

正確には（ホームページを作るために）WWWサーバの構築をする演習です。
次の3ステップからなります

1. WWWサーバをダウンロード
2. WWWサーバを起動
3. ブラウザで動作を確認

[例題] ホームページを作ろう(2): ダウンロード

- 各自の演習環境で、curlコマンドを利用し、次のPythonスクリプト(www.py)をダウンロードしてください
- 実行するコマンド
  - 具体的には `curl -O https://lpic-2024q2.demo.fml.org/dist/www.py`
  - `-O` は「マイナス」「大文字のオー」(OscarのO)

```
[コマンド]
$ curl -O https://lpic-2024q2.demo.fml.org/dist/www.py

[実行例]
admin@user99:~$ curl -O https://lpic-2024q2.demo.fml.org/dist/www.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2933  100  2933    0     0   3230      0 --:--:-- --:--:-- --:--:--  3226
```

(脚注) あとで curl の -o (小文字のオー)オプションも出てきます。 -O (大文字)は-o www.py (URLの右端にあるファイル名)の省略形

(アーカイブ動画で見ている人は)

一時停止して
手を動かしてみましょう

困っている人は「手をあげる」かチャットで合図を！ ブレイクアウトルームでTAさんが相談にのります

[例題] ホームページを作ろう(3): WWWサーバの起動

- WWWサーバを起動するには`sudo python3 www.py`コマンドを実行してください

```
[コマンド]
$ sudo python3 www.py

[実行例]
admin@user99:~$ sudo python3 www.py
/home/admin/www.py:26: DeprecationWarning: 'cgi' is deprecated ... (<-警告文)略 ...
(debug) serving at port 80
```

- デバッグメッセージ`(debug) serving at port 80`が表示されれば無事に起動しています
  - これは「WWWサーバが 80/tcp (TCPポートの80番)で待ち受けを始めました」という意味です
  - この演習はTCPなどの知識編を飛ばしているので、ここでは、そういうものだと思ってください

(脚注) www.pyには、さまざまな仕込みが入っていて、 初回起動時には自動的に簡単なホームページ(コンテンツ)を作成します

(アーカイブ動画で見ている人は)

一時停止して
手を動かしてみましょう

困っている人は「手をあげる」かチャットで合図を！ ブレイクアウトルームでTAさんが相談にのります

[例題] ホームページを作ろう(4): 動作確認

```
Welcome to my homepage
```

- 各自、手元のPCのブラウザでURL(`http://www.ユーザ名.demo.fml.org/`)にアクセスしてください
  上のように無事に表示されましたか？
  - URLの`ユーザ名`の部分を各自あたえられたID(例: user01)に差し替えてください
    例: user01 の人はURL(`http://www.user01.demo.fml.org/`)にアクセス
- [解説]
  - www.pyの初回起動時に /home/admin/htdocs/index.html というファイル(ホームページのコンテンツ)を自動作成します。 `Welcome to my homepage`というコンテンツは、このindex.htmlの中身です
  - 気になる人は、あとで、このファイルをエディタ等で開いて見てください

(脚注) 30年前の商用インターネット黎明期には、みんな、これだけで大喜びできたものです。 なにもかもみな懐かしい...

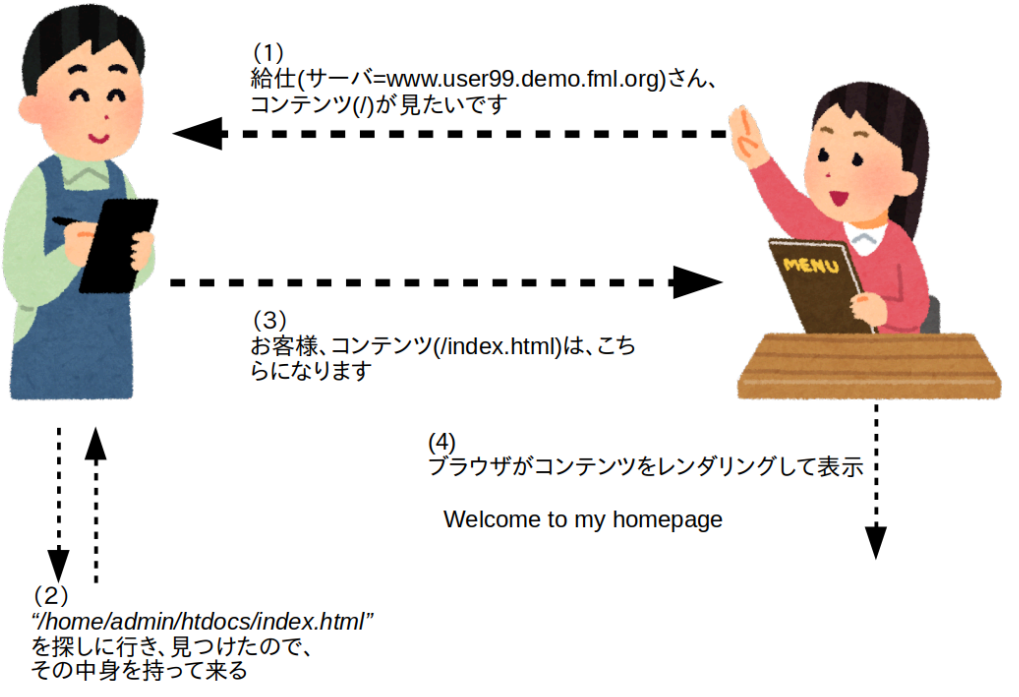
(アーカイブ動画で見ている人は)

一時停止して
手を動かしてみましょう

困っている人は「手をあげる」かチャットで合図を！ ブレイクアウトルームでTAさんが相談にのります

ブラウザの裏側を見てみましょう

- 右図は何頁か前に出てきた図です。
  この図の(1)(3)のところを、もっと生々しく体験してみることにします
- このあと二種類の方法で体験します
  1. curlコマンド
  2. telnetコマンド



[例題] curlコマンドでHTTPをしてみよう

- ブラウザが裏側で行っているデータ転送をコマンドで行ってみましょう

```
$ curl http://www.user99.demo.fml.org/
Welcome to my homepage
```

- [解説]
  - システム構築の際には、少し作業をしたら、こういった確認をして、少しずつ進めるべきなのです
  - curlコマンドを使うとWWWサーバの動作確認も簡単ですよね? システム構築の際の動作確認では、ブラウザではなく、こういったコマンドを使うと便利です。 ブラウザより圧倒的に軽いですから。 それに構築時の環境で、うまくブラウザを起動できるとは限りません

[例題] telnetコマンドで手動HTTPしてみよう(1)

- ブラウザが裏でやっていることを実感するために、 **生々しくHTTPを手動でやってみましょう**
  一緒にやります。字面だとタイミングがよくわからないでしょうから

- 手順

  1. まずtelnetコマンド「telnet 空白 サーバ名 空白 80」でサーバへ接続し、**TCPの通信路を作ります。**

     ```
     $ telnet www.user99.demo.fml.org 80
     ```

     サーバ名は各自ことなります。 右端の80がWWW(というサービス)を意味しています

  2. ここで次の1行をコピー＆ペーストし、ENTERを2回たたいてください(ここが**HTTP**)

     ```
     GET / HTTP/1.0
     ```

(脚注1) サーバの動作確認をする際に年中やります。 いままでに何百回やったことがあるかと...

[解説] telnetコマンドで手動HTTPするイメージ

「流しそうめん」に例えるなら

- (デジタル)「竹づつ」を用意するのが第一段階の「telnet サーバ 80」の実行で、
- 「そうめんが欲しい(流してください)」とリクエストするのが第２段階「GET / HTTP/1.0」に相当します(いわばGET SOMEN?)
  - リクエストの返事として、竹の上を流れてくる物がコンテンツ(そうめん)
  - コンテンツのやりとりの約束事がプロトコル (この例では HTTP)

(脚注) 竹づつの上でSSHプロトコルを使う約束も可能で、それがsshというアプリケーションです

[例題] telnetコマンドで手動HTTPしてみよう(2): 実行例

```
[実行例]
$ telnet www.user99.demo.fml.org 80
Trying 10.20.30.40...
Connected to 10.20.30.40.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.7.3
Date: Thu, 29 Feb 2024 23:02:03 GMT
Content-type: text/html
Content-Length: 23
Last-Modified: Thu, 29 Feb 2024 23:01:46 GMT

welcome to my homepage
Connection closed by foreign host.
```

(アーカイブ動画で見ている人は)

一時停止して
手を動かしてみましょう

困っている人は「手をあげる」かチャットで合図を！ ブレイクアウトルームでTAさんが相談にのります

wwwサーバとジャンケンしてみよう

[解説]ジャンケンの仕様

- コンピュータの中は数字で動いています。得意なのは数字だけとも言えます
- 逆に、数字で表現してあげれば、コンピュータは何でも出来ます
- ジャンケンを次のように定義してあげることでコンピュータとジャンケンが出来ます
  - グー = 0、チョキ = 1、パー = 2
  - 勝ち負け判定は「`3 + 自分の手 - 相手(コンピュータ)の手`を計算し、それを3で割った余り」
    あいこ = 0、(自分の)負け = 1、(自分の)勝ち = 2

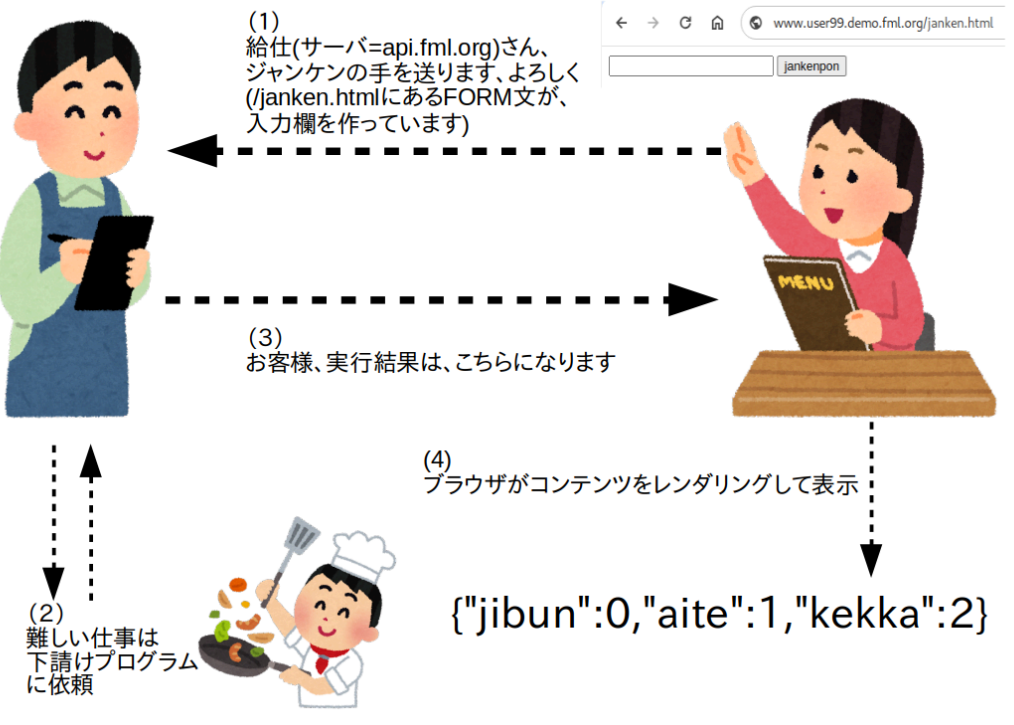
(脚注1) 仕様の紹介だけします。確かに、これでジャンケンになっていることを確認してみてください

(脚注2) 本ウエビナーではコーディングをしないので紹介だけですが、自由課題では必要な知識です

[解説]WWWサーバと会話する(値をWWWサーバに送る)(1)

- (1) URLで表示される入力欄(図の右上端を参照)に値を入れ、クリックすると、 自分の手(0か1か2)がサーバ(注:api.fml.org)に送られます
- (2) サーバで動いている”じゃんけんWeb APIサーバ”が、じゃんけんを行い
- (3) WWWサーバ(api.fml.org)がWWWブラウザに答えを返します

```
URL =
http://www.user99.demo.fml.org/janken.html
```

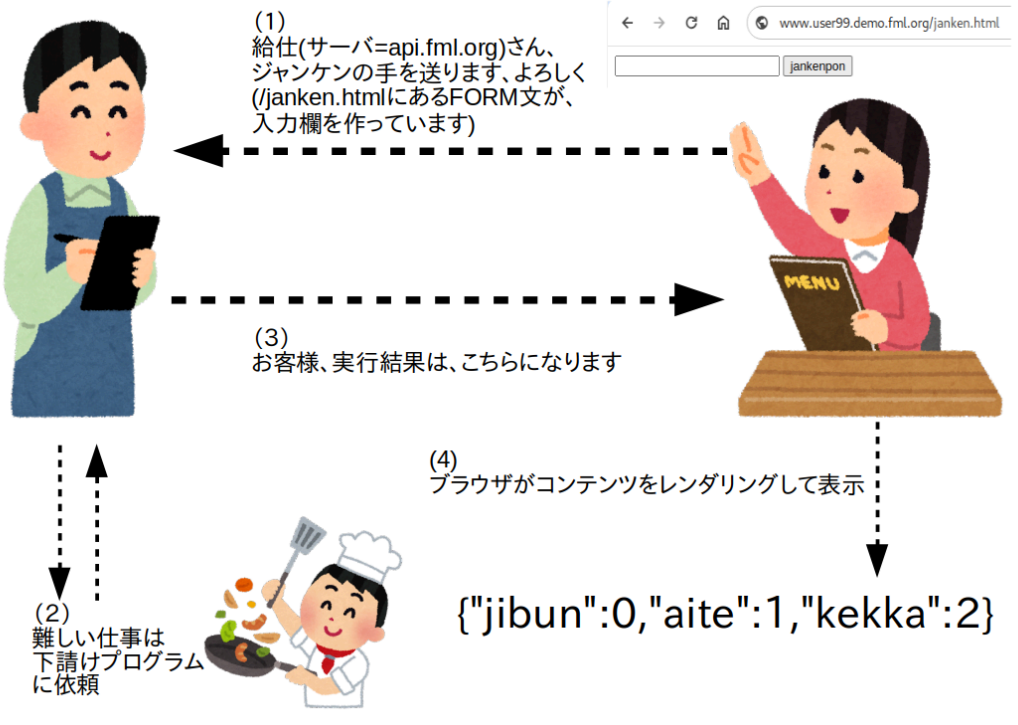


(脚注) 自分のPC、www.pyを動かしているサーバ、api.fml.orgという別のWWWサーバが登場しています。 区別がついていますか？

[解説]WWWサーバと会話する(値をWWWサーバに送る)(2)

- **注意**:janken.htmlにはform文というHTMLが書いてあります(次頁を参照)。 form文にあるactionにはジャンケン(Web API)サーバのURLを設定しますが、 ここに**api.fml.org(www.pyとは違うサーバ)**名を指定します(ポイント)
  - actionのURL右側(`/api/janken/v1`)はコンテンツの場所ではなく”じゃんけんWeb APIサーバ”の呼び出しと解釈されます(そのようにWWWサーバを作っているのです)

```
<form ... action="http://api.fml.org/api/janken/v1">
```

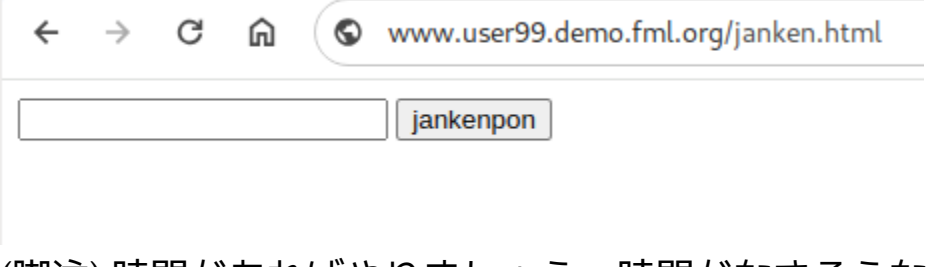


(脚注) 自分のPC、www.pyを動かしているサーバ、api.fml.orgという別のWWWサーバが登場しています。 区別がついていますか？

[解説]janken.htmlのFORM文(HTMLで値をWWWサーバに送る)

```
<form method="POST" action="http://api.fml.org/api/janken/v1">
  <input name="jibun"  type="text"/>
  <input type="submit" value="jankenpon">
</form>
```

- このあとダウンロードするjanken.htmlの解説です
- ブラウザには、この命令(`<form>`や`<input>`など)を解釈して画面を作る機能があります
- `<input name="jibun" type="text"/>`により**入力欄**が作成され、`<input type="submit" value="jankenpon">`により**「jankenpon」**のクリックボタンが作成されます。そして**「jankenpon」**をクリックすると(**入力欄**に入力された)値を(actionで指定した)サーバへ送信する機能もブラウザにあります
- 値を送信する先は`action=URL`で指定するURLです



(脚注) 時間があればやりましょう。時間がなさそうなら飛ばします

[例題] WWWサーバとジャンケンしてみよう(1) 手順

- 作業手順は次のとおりです

1. www.pyとは別のターミナルで作業します。ターミナルその2に移動してください
2. curlコマンドでjanken.htmlをダウンロードします
   - 具体的には/home/admin/htdocs/以下にjanken.htmlをダウンロードします
3. ブラウザでアクセスし、ジャンケンの手を入れ、「jankenpon」をクリック
   - アクセス先は、user99の人は`http://www.user99.demo.fml.org/janken.html`
     注: user99の部分は各自となります。与えられたIDと差し替えてください
   - ジャンケンは手を数字で入力してください。仕様: 0 = ぐー、1 = ちょき、 2 = ぱー

(脚注1) Ctrl-Cは「CtrlとCのキーを同時に押す」操作です。 実行中のコマンドを途中で止める時の定番 (脚注2) 本演習はwww.pyを編集しないので、 Ctrl-Cで止めなくてもよいのですが、バックグラウンド処理など別の解説が増えるので、単純な方を選択しています

[例題] WWWサーバとジャンケンしてみよう(2) ダウンロード

```
[コマンド]
$ curl -o htdocs/janken.html https://lpic-2024q2.demo.fml.org/dist/janken.html

[実行例]
$ curl -o htdocs/janken.html https://lpic-2024q2.demo.fml.org/dist/janken.html
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2933  100  2933    0     0   3230      0 --:--:-- --:--:-- --:--:--  3226
```

- `-o`は「マイナス」「小文字のオー(OscarのO)」です。引数でファイル名を指定できるオプションです
- sshでログインすると`/home/admin`という場所にいます。 `curl`コマンドを`-o htdocs/janken.html`オプションつきで実行すれば、 `/home/admin/htdocs/janken.html`にファイルを直接ダウンロードできます

(脚注) 階層構造が希薄なスマートフォンとかWindowsのせいなのか、 引数に/(スラッシュ)が含まれると、とたんに分からなくなる人が多いようです。 冷静に文字列の足し算をして考えてみてください。 慣れるには場数しかないです

[例題] WWWサーバとジャンケンしてみよう(3) 動作確認

1. URLにアクセスし(注:user99の部分は差し替え)
2. 入力欄に 0 (グー)か 1 (チョキ)か 2 (パー)を入力
3. 「jankenpon」をクリックすると
4. 右図(下)のような結果(JSON形式)が帰ります
    - `jibun:0` が自分の手
    - `aite:2` はコンピュータ(相手)が考えた手
    - `kekka:1` はジャンケンの勝敗です
        - 0(あいこ)、1(自分の負け)、2(自分の勝ち)

```
URL =
http://www.user99.demo.fml.org/janken.html
```

(脚注) JSON形式とはJavascriptで使われるデータ形式で、見かけはPythonのdictと同じです。{ KEY1:VALUE1, KEY2:VALUE2, ... }

返事がJSON形式になっている理由は、これを利用したWeb API開発をする自由課題のための仕こみだからです

(アーカイブ動画で見ている人は)

一時停止して
手を動かしてみましょう

困っている人は「手をあげる」かチャットで合図を！ ブレイクアウトルームでTAさんが相談にのります

[例題] ジャンケンWeb APIサーバにtelnetしてジャンケンしよう

前頁のFORM文の「jankenpon」をクリックした際にブラウザが裏で行う動作を実感しましょう

- api.fml.orgに接続し、

```
$ telnet api.fml.org 80
```

- 以下の呪文をコピー＆ペーストしてみましょう

```
POST /api/janken/v1 HTTP/1.0
Host: api.fml.org
Content-Length: 7

jibun=0
```

(脚注) ブラウザがFORM文を解釈して、送信する内容が、このコピー＆ペーストする文字列です。 意外と簡単ですよね？

[例題] ジャンケンWeb APIサーバにtelnetしてジャンケンしよう

- [動作確認] 次のような結果が返ってきましたか？

```
{"jibun":1,"aite":0,"kekka":1}
```

- （数字は異なるでしょうが）同じものがブラウザでも見えていたはずです
- モダンで素敵なウエブページも（ブラウザの裏側で）これを何度も何度も行っているだけです。いまどきのウエブページはページを構成する部品が多いので、ひとつのページで何十回もやっていたりします

(アーカイブ動画で見ている人は)

一時停止して
手を動かしてみましょう

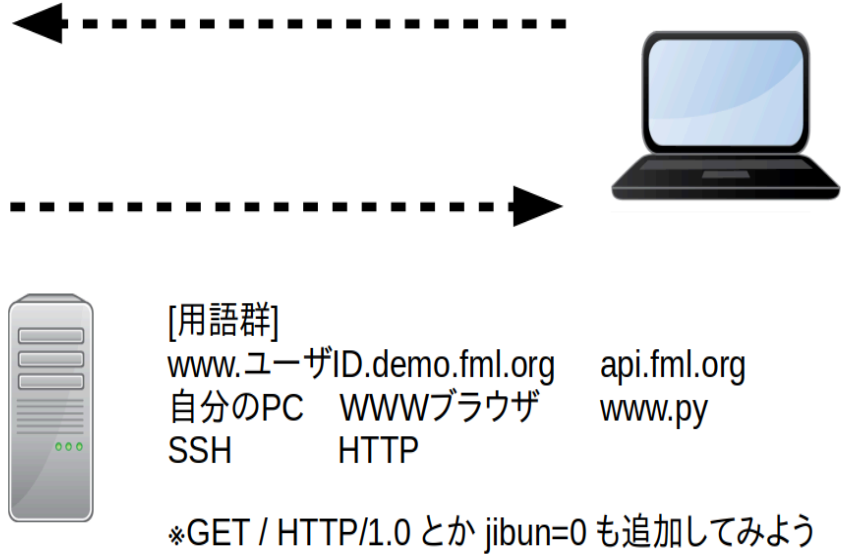
困っている人は「手をあげる」かチャットで合図を！ ブレイクアウトルームでTAさんが相談にのります

[例題] 構成図を書いてみよう

Q: 配布する構成図テンプレートの、どこに何が当てはまるか考えてみてください

- 図の右下にある用語群を書き入れてください。てきぎ機材や矢印の増減もしてください。 なお、図を描くツールは自由です(手書きでもOK)
- 図のテンプレートはこちらから
- 簡単な答えあわせをイベントの最後に行います (長い版は別動画を用意するので、そちらを参照)
    - すこし時間が必要でしょう。 このあと、少しキャリア形成関連のコラムのような話が続きます。 それを聞きながらでよいので考えてみてください

構成図テンプレート



(脚注) 最後は、本演習の総復習をかねた構成図を書く演習です。 あんがい、どこでどう動いているか混乱していませんか？

おわりに

ハンズオンは、いかがだったでしょうか？

- もう少しアレコレしたいと興味が持てた人は、この業界に向いているかもしれません
- 30分程度のITインフラ体験をしてもらいましたが、本当に基礎の体験だけでした
  - (短時間なので出来ませんでしたが) 手順書に沿った作業ではなく手順を自分で考える訓練が望ましいです。 また、いま何故こういった技術の形に落ち着いているのか？も疑問にもって欲しいです
  - 理由は、このあと少し話します
- もう少しアレコレしたい方へ
  - 物足りない人むけに、付録A.自由課題が用意してあるので挑戦してみてください
  - また、本日つかっている演習環境は、このあと壊してしまうので、 別途、自習用(演習環境)の手配が必要です。それについては付録B.を参照してください

(脚注) 授業では、参加任意の加点対象枠として「自由課題」を出題しています

[私見] 長く続けられる技術者像とは?(半分は再掲)

1. ひととおり技術が何でも分かる(再掲)
    - 得意分野としてITインフラいかがですか？
    - 得意というのは、~~「クラウドの使い方が分かる」ではなく、~~ 動作原理が分かるレベル
2. 論理的能力
3. 技術の先読み(?)

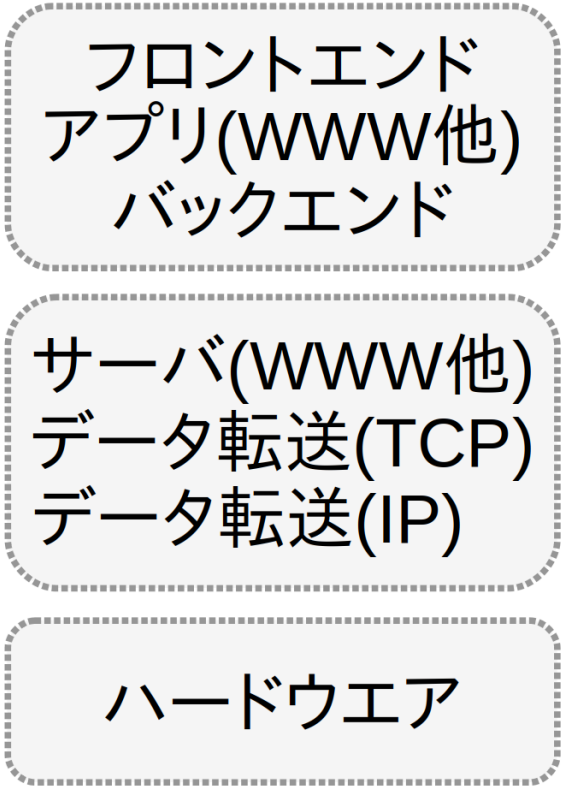


図: かなり大味のイメージ

(脚注) 異論は、あるでしょう:-) 次頁以降Q&A形式で少し語りたいと思います

Q: ITインフラ分からなくても大丈夫では？クラウド全盛なんだし

- 確かに一見すると、クラウドは簡単に使えそうです
- しかしながら少し凝った設定をしようとすれば生のUnix/Linuxコマンドが顔を見せます
- そのため、Unixコマンドや/etcの下の設定ファイルの編集方法が分からないといけません
- それに、どの分野でもITインフラは分かったほうがいいみたいですよ?
  - 例: ニコ生Webフロントエンドチーム の業務内容をみると、かなりの(8割がた?は)インフラ系

(脚注1) もっと重要な脅威があります。次頁へつづく

Q: クラウドサービスの脅威と厳しい現実とは？

- 大手のクラウドサービスは**世界中どこで誰が使おうと同じです**
  - だから**使い方が分かるだけでは差別化になりません**
- クラウドだけでなく**「流行の技術」は、もっと多数のライバルがいる**でしょう
- 自分のウリは何か？ つまり自分の価値を差別化するには？
  - (悲しいかな、興味を持ってくれる人が少ないため)ITインフラが分かると、とても重宝されます
  - そこで、ITインフラに詳しくなってみませんか?(逆張りの勧誘:-)

(脚注1) プログラミング言語は世界共通ですし、ITインフラは管理画面の言語が違うだけです。 同じスキルのライバル同士なら低価格の人が受注できます。 いままでは 日本の企業文化や日本語の壁でライバルが参入しづらかった面がありましたが、これからは?

(脚注2) 流行の技術や資格は就職に役立つとは思います。でも10年後も食べていけると思いますか？

(脚注3) ITインフラ産業は不況に強いし、不滅です。 いまや24時間動いていて当たり前のインフラになったし、 24時間動かしつづける使命感で、この産業をやっている人も多いでしょう。 ただ(dog yearと言われるように約7倍速で)移り変わる技術を追う努力は必要です。 今なら「プログラマブルなITインフラが分かること」でしょうか？「Ciscoのシリアルから直接設定が大好き」では、もうダメ;)

Q: 論理的能力よりコミュ力ではないのですか？

- もちろん、コミュニケーション力がとても大事な職種(例:営業)もあるでしょう
- しかしながら、商談の発掘のために、お客様と技術部が会う必要はないはずです
- お客様の話を論理的に分解・再構成して整理・提案できるのが技術のプロでしょう
  - お客様の曖昧な話から、本当にやりたいことを推測して、 それを論理的に再構成・整理・提案できるのがプロであろうと思うわけです (それができないなら、わざわざ専門家を呼ぶ理由がない)
  - 本ウエビナーの題材を例にとれば、 お客様は「ホームページを立てたい」と言えばよいだけなのです。 そこからプロジェクト一式を仕切れるのがマネージャでしょうが、 まずは「WWWサーバを構築」と言われたときに、おまかな手順書を、すぐに書けるくらいを目指しましょう (3年?)

(脚注) ITインフラって、しゅくしゅくと地道な作業が好きな人に、とても向いていると思っています。 もっとも引きこもりではダメですが... お客様と対面で1,2回ミーティングするくらいは我慢しましょう:-)

Q: 奇跡が努力の先にあるなら？…どういう修行をするべきですか？

(1) **ヒマがあれば手を動かす**

(2) **アウトプットする**

(3) (即効性はありませんが)歴史の勉強もしよう

(脚注) 響け！ユーフォニアム(京アニ)は名作;文科系スポ根のBildungsromanいいですよね〜; S3放映中(注:本動画は2024年春作成)

ヒマがあれば手を動かす

- 努力は、それなりに酬われると思います
  - 努力すればIT業界のサリエリには、なれるかもです
  - 芸術系(?)とは違いますから... (サリエリ100人あつめても、モーツァルトは育たないですよね?)
- やっぱり場数なんですよ
  - 「練習で出来ないことは、本番では絶対に出来ません」
  - 手を動かすなら、OSのパッケージを増減して遊ぶより、 ソースからコンパイルするとか、 OSのカスタマイズ(例:1.4MBのLinux作るとか(OSの実務的な動作原理の理解が必要なこと))に挑戦したい

(脚注1) 1.4MBはフロッピーのサイズ (脚注2) OSの動作原理というとカーネルの話と思われそうなので「実務的な動作原理の理解」と書いています。 必要な知識は、カーネルの作り(例: スケジューラや仮想記憶)ではなく、 Unixユーザランドを管理しているプログラム群とは？ どれが必須なプログラムなのか？ どのように起動してくるのか？ などです。 これらが分からないとOSを削っていくことが出来ません。 カーネル自体も削りますが、基本的にデバイスを削るだけなので、OSの授業でやるような内容は分からなくてもOKです (脚注3)映画「アマデウス(1984)」 (脚注4)「はちみつとクローバー」の方が良い例?天然物の天才はぐちゃんは次元が違うでしょ？

アウトプットする

- **Technical Writingの練習が必要です、読書感想文ではなく論理的な文章の書き方**
- これも場数なので、書くしかないです
  - 書く動機づけに、定期的にblogを書くとか技術同人誌を作るとか、心に決めて「〆切駆動開発」とか「イベント駆動開発」(イベントあわせで頑張ること:-)
  - twitterも140文字に文章を要約する訓練として使うなら、とてもよい使い方でしょう
- 1年後には、この教材で教える側になろう！と目標をたてるのもよいですね

(脚注1) Technical Writingは小学生からやるべきなのに日本ではやらない。 それを自覚して自主努力しないと書けるようにならない

(脚注2) わたくしも、 いちおう商業出版物で100万文字くらいは書いてると思います

歴史の勉強と、技術を先読みできるためには？

- (即効性はありませんが)目利きとか先読みには大事なところではないかと
- 歴史は繰り返しませんが、似たようなところには返ってきます(スパイラル?)
    - 世界は非合理だし目的論でもない。 出来事や英雄の歴史ではなく、 その背後にある**技術開発をドライブした観念や妄想や偶然や非合理なもの**の理解が重要と思うのです
- まとまった資料は？と言われても？... ないから自分でちまちま語ってます
    - シリーズ「色づかないWorld-Wide-Webの昨日から」 (qiita版は、かなり薄味です)
    - 濃い方はコチラ -> フリーソフトウエア運動の40年 (序章部分) (第1章)

(脚注1) スパイラル感といえば、 商用インターネット初期より今の方がメインフレーム時代に戻ってきた感がありませんか？

(脚注2) 商用インターネットがメジャーになったあと(1996あたり以降)に書かれた(人文科学系の視点からの)書籍、 出来事史、細部がアテにならない回想録とかは、それなりにありますが、なんか違うような...

妄想(Vision?)は世界を変えたか？YES！たとえば

- Vannevar Bush
- Douglas Engelbart
- Stewart Brand
- Steve Jobs

(脚注) 申し訳ないけれど、ネタをふるだけふって終わりです。 たぶん1Q(7-8週)くらい授業しないと話おわりません

余談ですが、来月(2024/07)の 色づかないWorld-Wide-Webの昨日からは、 V.Bushを予定しています

まとめ

- 気になるフタは開けよう
- 論理能力をきたえるには？場数ですね
- 面白い気がしたのなら、ぜひITインフラストラクチャ、おすすめ

質問コーナーです、どうぞ

構成図の答えあわせ

付録

付録 一覧

- A. 自由課題
- B. 自由課題を行う演習環境の作り方
- C. リファレンス
  - 本イベントの配布物サイトや元ネタへのリンク
- D. 本コンテンツを利用したい方々への御提案
- E. AWS Academyの御紹介

付録A. 自由課題

- 100番台 ... やさしい
- 500番台 ... 若干むずかしい
- 900番台 ... かなりむずかしい(学部４年ないし大学院レベル)

(脚注) 自由課題、ひと月ほど時間をあげるので考えて見てください。 解答編の動画を作るかは、リクエスト数しだい？

自由課題(1): やさしい

- 問110: index.htmlを編集し、もっと素敵なホームページを作成してください
  - 例: ホームページを `Welcome to 自分の名前` にしてみましょう
  - 例: もっとリッチなホームページにしてみましょう。やりかたは検索すると色々みつかります
- 問120: ISBNで書誌情報を検索できるページを作成してください
  - janken.htmlを参考にisbn.htmlを作成するとよいでしょう
  - 使うサービスはopenbd.jp。 openbd.jpには説明がありませんがPOSTメソッドでも検索可能です

自由課題(2): 若干むずかしい

- 問510: www.pyを改造してジャンケンWeb API機能を実装してください
  - 動作確認は、janken.htmlを編集してactionのURLを自分のwww.pyを動かしているサーバに向けます。これで、ジャンケンができるか？を確認します。 www.pyのログに自分がアクセスした様子が出ることを確認してください。
  - www.pyの中に`janken`メソッドの雛形があるので、そこに実装します。 `/api/janken/v1`の場合に`janken`メソッドを呼び出すルーティングは、 lsformメソッドを参考にしてみてください
- 問520: 演習で体験した初期のHTTPの仕組みでは、ソフトウエアの配布は出来ても、 商用ショッピングサイトは作れません。 そのために、どのような技術が開発されてきたのか？を調べてみましょう。
  具体的には
  - ログインなどの「状態のあるシステム」を、どう実現すればよいでしょうか？
  - 個人情報やクレジットカードを送信する際には暗号化してほしいですよね？

(脚注1) 問510は数行のPythonプログラミングです。 この方向の課題は色々かんがえられますが、 HTTPの裏側を知るというテーマから逸脱していってしまうので1題だけ入れておきました (脚注2) ハンズオンでは30年前のインターネットを体験しました。その後HTTPには様々な改良が加えられています。 問520は、その後の30年の歴史の概略を問う課題です

自由課題(3): 若干むずかしい

- 問530: 演習では手間を省くために自作のwww.pyを使っていますが、 実際の運用では、定番のWWWサーバを使いウエブサイトを構築します。 定番中の定番、apache(発音：あぱっち)でホームページを立ててみましょう
  - index.htmlは同じものを使ってください
  - 設定ファイルを編集する必要があります(ここは各自で調べてください)
- 問540: 問530と同じ課題をnginx(発音：えんじんえっくす)で、やってみてください

(脚注) インストール自体は、ただapt installするだけなので「やさしい」のですが、 各ソフトウエアの設定ファイルが、それぞれ異なる場所にあり、異なる流儀で書かれているため、 初心者には難しいらしいです。 それで「若干むずかしい」に分類しました。 でも流儀なんて数系統しかないので、そのうち慣れます。いや、この際なれましょう:-)

自由課題(4): だいぶ難しい

ハンズオンでは経験しづらいのですが、HTTPの実際のデータ転送はTCP/IPが行います。 TCP/IPはインターネットの基本的な動作原理で、われわれはTCPを40年以上つかってきましたが、 いまHTTPは「TCPではない仕組みのHTTP/3」へ移行しつつあります。

- 問910: HTTP/3の動作原理を調べて見てください。それ以前のHTTP/2と何が違うのでしょうか？
- 問920: HTTP/3のメリットとデメリットとは何でしょうか？
- 問930: オンラインショッピングのサイトを運用しているあなたは、 そのサーバをHTTP/3化するべきでしょうか？ HTTP/3化する場合、どのような設計にするべきかを考えてみてください

(脚注1) 1983年1月1日にTCP/IPへ全面的に切り替わりました(第1世代のNCPの利用が終了)。それから41年です

(脚注2) 1983年の夏の終わりには、TCP/IPを搭載した(いわばARPANETの推奨OS)4.2BSD(バークレイUNIX)が公式リリースされました

(脚注3) 難易度の想定は、学部4年以上です

付録B. 自由課題を行う演習環境をどうするか？

- Windows11だけが大問題です。 素のWindowsで頑張ろうなどと変なことを考えるとハマるので、 すなおにDockerを入れて、この演習環境を起動するのがよいとおもいます。
- こうすれば、Windows、MacOSX、そしてLinuxの人、みなさん同じように演習できるはずです
- Docker環境を用意する手順
  - 演習環境のコンテナイメージをdocker hubに公開しておきました。 こちら -> fmlorg/debian-pc
  - 作業手順は3段階で
    - (1) docker CLIもしくはDocker desktopをインストールする
    - (2) ターミナルでdockerコマンドを使い演習環境(コンテナ)をセットアップ
    - (3) 演習環境にログインして演習を行う
  - 補足: Windowsでも、 Docker desktopのインストール(1)が終われば、 ターミナル(コマンドプロンプトやPowershell)でdockerコマンドも使えます
  - 次頁以降では Windows11 の環境構築の実行例を紹介します

(脚注1) 付録B.はdockerが分かる前提です。 この際なので、自由課題としてdockerを勉強してください:-) (脚注2) 無料のPythonプログラミング演習環境Webサービスなら色々ありますが、 インターネットから80/tcpにアクセスできる無料のサービスは見つけられませんでした。 まぁ中途半端なWebサービスを探すより月額500円の低スペックVPSでサーバたてましょう！ それがおすすめ

(1) Docker desktopのインストール

- https://www.docker.com/products/docker-desktop/
- ページ上部メニューの Products から Docker Desktop を選択してクリック
- Download for Windowsをクリックしてインストーラをダウンロード
- ダウンロードフォルダにDocker Desktop Installer.exeというファイルがあるはずです。
  それをクリックしてインストールしてください

(2) Powershellでdockerコマンドを使う(実行例)

[実行例(コンテナの起動)]

```
PS C:\Users\user99> docker run --rm -d -p 80:80 --name ex fmlorg/debian-pc
 ... 略 ...
ceb335b95d58aac1dcf57f2df54d05e6c05473cb382e75c79edc6195ad910422
```

- 上のように docker run すれば、 本ウエビナーで使っていた演習環境と同じもの(コンテナ)を起動できます
- `--name ex` でコンテナに名前 `ex` (exerciseのex)をつけています。 次ページでもdockerコマンドの引数に `ex` が使われていることに着目してください

(脚注) Docker Desktopのインストールが終わればdockerコマンドが使えるようになっています。

このページ以降では、ターミナルでdockerコマンドを叩きます

(3) Powershellから演習環境にログインする(実行例)

- コンテナへはsshではなくターミナルの上で直接はいります。この場合パスワードもありません。
- debian-pcコンテナに入ると、ユーザrootの立場でコンテナの中に出現してしまいます。
  演習はユーザadminで行うので、ユーザadminになり代わってから演習を行ってください
  - suコマンドでユーザadminに切り替えます (注: su = substitute user)
  - 場所が異なるので、引数なしのcdコマンドを実行してください。これで/home/adminへ移動します
  - これでテキストと同じ演習環境になりました

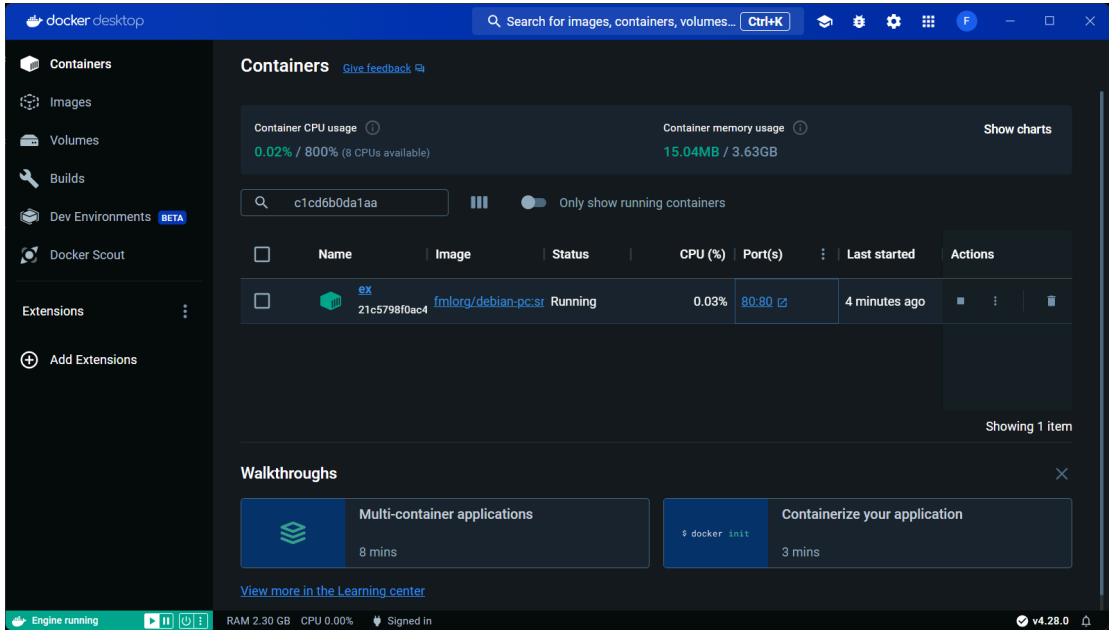
```
[実行例(コンテナ(コンテナ名はex)へログインする)]

PS C:\Users\user99> docker exec -it ex /bin/bash
root@ceb335b95d58:/# su admin
admin@ceb335b95d58$ cd
admin@ceb335b95d58$ curl -O https://lpic-2024q2.demo.fml.org/dist/www.py
  ... 省略 ...
admin@ceb335b95d58$ sudo python3 www.py
(debug) serving at port 80
```

(脚注) 直接とは? -> 「目の前でPCを操作できる危ない状態」の仮想版くらいで納得してもらえれば、まぁ、はい。

(3) docker desktop (実行例)

- 動作確認するには、 中央あたりにあるポートマッピング `80:80` をクリックしてください。 コンテナの中で実行しているwww.pyにHTTPでアクセスできます

- この環境では、WWWサーバにドメイン名をつけられません
  - docker desktopからクリックするか
  - `http://localhost:80/` へのアクセスで確認してください (あえて言えばサーバ名がlocalhost)



付録C. リファレンス

- 演習で使うスクリプトやワークシートは、こちら(https://lpic-2024q2.demo.fml.org/)で配布しています
- 本日のスライド (lectures.fml.org/slides/skill-exp/lpic-2024q2/)
- 元になった演習(1.と2.)と、1.の大元の演習(3.); いずれも情報システム工学科3年生の授業

1. 情報システム開発基礎演習 (必修, CC BY-NC-SA 4.0)
2. コンピュータネットワーク (必修, CC BY-NC-SA 4.0)
3. クラウドコンピューティング (選択, CC BY-NC-SA 4.0)

付録D. 本コンテンツを利用したい方々への御提案

- インターンシップや新人研修の初日用にお使いいただけるのではないか？と考えています
  - 実際には**1/4倍速**くらい、 つまり1日くらいかけて、ゆっくりやるのがおすすめです
- 「超初心者向け」の想定です(スーパービギナーどころかハイパービギナー編)
- 十分な数のアシスタント(TA/SA)は必須です
- 動画作成に合わせて、演習の環境構築方法と指導要領をまとめた資料(技術同人誌?)を作りました。 ご入用の方は御連絡ください

付録E. AWS Academyの御紹介

PowerPoint プレゼンテーション　1 / 1　100%

このチラシ(Flyer(PDF))の高解像度版のダウンロードは こちら からどうぞ。なお情報は2024年5月時点のものです