情報科教育法II　2002＃5

久野　靖*

2002.11.13

はじめに

前回は皆様に「プレゼンテーションの発表」をやって頂いたので、講義をやっている時間がほとんどありませんでした。が、頂いた感想を見ると「やってみてはじめて分かった」という主旨のものが多かったのでよかったと思います。逆に言えば、皆様が高校生にプレゼンテーションを教えるとしたら、やっぱり (今回のような方式がいいかは状況によるでしょうが) 何らかの方法で「全員が 1 回は喋る体験をする」ことが必要ですよね。その方法として、「向かい合って 2 人 1 組でやる」というのは一番短時間で済む方法であることは確かです。

ただ、聞き手が一人というのは淋しいですから、時間的余裕があれば 3〜4 人でグループになって順番に発表し (これに 1 時限)、次の 1 時限ではできのいい人が数名、クラス全体の前で発表する、とかがいいでしょう。「10 分」というのは練習用の発表としてはよい長さだったでしょう?

さて、今回の内容ですが、「情報」の先生になるなら絶対に理解しておくべき、ネットワークの原理とセキュリティについて取り上げておきたいと思います。

- ネットワークの原理
- 模擬授業「情報 C」p.112-113
- ネットワークの安全性とは?
- HTML 教室 — スタイルシート再び

1　ネットワークの原理

1.1　インターネット

そもそも、我々がブラウザで海外のあちこちのサーバから情報を取り寄せているとき、具体的にはどうやって通信が起きているのだろうか?

a. 手元 PC →大学内 LAN →プロバイダ→電話会社→国際電話接続→電話会社→相手側プロバイダ→相手組織 LAN →相手側サーバ

b. 手元 PC →大学内 LAN →プロバイダ→インターネット日本支部→米国本部→相手国支部→相手側プロバイダ→相手組織 LAN →相手側サーバ

c. 手元 PC →大学内 LAN →プロバイダ→プロバイダ間相互接続→次のプロバイダ→…(くりかえし)…→相手側プロバイダ→相手組織 LAN →相手側サーバ

*筑波大学大学院経営システム科学専攻

一番正解に近いのは c。インターネットの特徴は、多数のネットワークの融合体であること。ここで「プロバイダ」となっているものは代わりにネットワークプロジェクトや公共ネットワーク (大学間ネットその他) などである場合もある。

では次に、上のような経路でたとえば www.yahoo.com にネットワーク接続をするとき、どんなことが起こっていると思うか?

a. まず u-gakugei.ac.jp のノードに (中継点) に行き、次に ac.jp のノードに行き、次に jp のノード、そこから com のノード、yahoo.com のノードとたどっていって相手サーバにつながる。

b. まず www.yahoo.com を検索により番号だけから成るアドレスに変換し、長距離電話のように相手サーバにダイヤルして各中継点ごとに回線を確保して相手までの経路ぜんぶを予約し接続する。接続が終わったらその上でデータ転送を行い、最後に切断する。

c. まず番号だけから成るアドレスに変換するのは同じだが、「接続開始」「データ要求」「結果返送 1」「結果返送 2」…「終了」などのメッセージはバラバラに、それぞれ「送り主」「宛先」が先頭にくっついた形で送られて行く。

これも正解は c. まず、www.u-gakugei.ac.jp などの「長い名前」(ドメイン名) は人間にはよいがそのままでは計算機にとっては扱いづらいので、固定長 (32 ビット) のアドレス (IP アドレス) に変換され、以後はそれを用いて処理する。この、アドレス変換を行う部分 (DNS — Domain Name System) では、ちょうど a. に書いてあるようにドメイン名の各部分ごとに担当するサーバがあって変換表を管理するようになっている (「情報科教育法」p.136-137 を読んでおいてください)。

次に b. と c. の違いだが、b. のように通信開始時に経路を確保する方法は「回線交換」といい、電話などで使われ来た。しかし、非常に多数のユーザがバラバラに大量のデータを交換するインターネットではこれは向いていない (ネットを使おうとしたら「お話中」では嫌でしょ?)。なので、c. のようにデータはバラバラのまま他の人の通信内容と混ざりつつ個別に送られて行く。このバラバラの単位を「パケット」(小包) といい、このような方式全体を「パケット交換」という。32 ビットのアドレスだけでそれぞれの「枝分かれ」点で正しい方向に行けるように、枝分かれ点に設置されている装置 (ルータ) が頑張って処理をする (「情報科教育法」p.134-135 を読んでおいてください)。

パケットの大きさやパケットにアドレスやデータをどのような形で詰め込み、やりとする「約束ごと」(通信規約 — プロトコル) は、IP(Internet Protocol) と呼ばれている。

インターネットの特色は、この「パケット交換を用い」「多数のネットワークどうしが相互につながった形で、特定の中央組織なしに運用され」「共通の IP アドレスと IP プロトコル (通信規約) を使用する」ところにある。これらの約束に従えばどこのどのマシンでもインターネットを通じて世界中のインターネットにつながった任意のマシンと通信できる。

1.2 プロトコル階層

ネットワークによる通信は多数の「約束ごと」(規格、標準) から成り立っている。これを分かりやすく扱うため、「層」(layer) に分けて考える。

一番下のレベル (物理層) では、ケーブルの規格とか、コネクタの形状の形とか、使用する電圧とかを定める規格がある。これらが合っていない通信できないのは当然。

次のレベル (データリンク層) では、1 つのパケットを隣りの機器どうしでやりとりするための約束ごとが決められている。たとえば多数の PC が同時に送信しようとしたとき、本当に同時だと信号が混ざってしまうので 1 つずつ順番に送るよう調停するなどの方法も決められている。

その上に IP の約束ごとがある。このレベル (ネットワーク層) では、パケットの行き先を制御して正しく送り先に届けるための機能が中心になっている。

ここでちょっとクイズ。ネットワークが混雑してある中継点 (ルータ) に大量のパケットが殺到してさばき切れなくなったらどうすると思う?

a. あふれたパケットは黙って捨ててしまう。

b. 混雑しているから送信量を減らしてくれという特別なパケットを送る。

c. N 本の線が集まるならその N 本の線から最大量の送信が来ても大丈夫なような処理能力を持たせるのであふれることはない。

正解は a. そもそも b. みたいなことをしても現に殺到しているパケットが処理できるようになるわけではない(混雑していることを上流に教えるという効果はあるにせよ)。c. はそもそもウソでしょ? N 本の線のうち、N-1 本がすべて残り 1 本宛てのパケットを最大能力で送って来たら絶対にあふれる。機器の処理能力の問題ではない。

そういうわけで、インターネットは「時々エラーや混雑があってパケットが捨てられてしまっても大丈夫」なように作られている。これは、インターネットが米国防総省の予算で「核戦争が起きてあちこちの機器が破壊されても残った機器だけで通信が続けられるネットワーク」の研究としてはじまったことにもよっている。

しかし、パケットが捨てられても大丈夫なようにするって、どうするの? それは、パケットを送信する側で、送るパケットに順番に番号を降っておく。そして受け取り側である番号のパケットが到着しないようなら、その番号のパケットを送り直してもらうようにすればよい。でも、「これが来ないから送り直してくれ」という返事パケットが捨てられてしまったらどうするの???

これに対処するため、実際には次のようになっている。

- 送り側がパケットを送ると、受け側では「何番まで正しく受け取った」という返事を返す。
- 受け側は一定時間内の返事が帰ってこないなら、その番号のパケット以降を再送する。

だから、「返事」のパケットが失われた場合は、送り側は対応するパケットを再送する。しかし受け取り側はパケットについている一連番号を見れば既に受け取り済みかどうか分かるので、2 重に到着したものは単に無視するだけでよい。この方法で「エラーのない通信」機能を提供するレベルをトランスポート層と呼ぶ。

トランスポート層の上のレベルには、個別のさまざまなプログラムないし用途専用の複数のプロトコルに分かれている。たとえば:

- ftp — ファイル転送プロトコル
- http — Web ページ取得用プロトコル
- smtp — 電子メール配送用プロトコル
- pop — メールサーバから手元マシンへのメール取り寄せプロトコル

などなど。たとえば UNIX 上の `telnet` コマンドを使えば一部のプロトコルを手で打ち込んで試してみることもできる。

2 模擬授業

今回の模擬授業は「情報 C」p.112-113「情報システムの安全性」です。先生役の人、よろしく。

3 ネットワークとセキュリティ

3.1 セキュリティとは?

セキュリティ(security、安全性)って何ですか?
セキュリティに関連する概念:

- 信頼性 (reliability) — 機器等の故障の少なさ

- 可用性 (availability) — いつでもちゃんと使える状態にあるかどうか
- 依存可能性 (dependability) — 機器にたよって仕事をしてもいいかどうか。たとえば動いているけどデータがたまに壊れているとかだと依存するわけには行かない。
- 危険のなさ (safety) — その機器の使用によって危害を受ける度合の少なさ (感電するとか…)

セキュリティといった場合、「意図的な/外部からの攻撃に防御されている度合い」というニュアンスが強い。安全性というより堅牢性とか言う方がいいかも知れない。

たとえば、地震や火事などの災害に対して防御すること、泥棒にマシンごと盗まれないようにすることもセキュリティ対策の一貫。(例: バックアップを取るとか鎖でつないでおくとか。) ただ、この手の対策は計算機システムに固有のことではないし、まっとうな企業ならどこでもやっている。

では計算機固有の問題は何かというと…

- 既に学んで来たように、現代社会では情報そのものに大きな価値
- 情報が盗まれたり破壊されることは巨大な損害につながり得る
- しかし情報が盗まれたり破壊されても見た目わからない
- しかも物理的に侵入されなくてもネットワーク経由でやられるかも

3.2 さまざまな場面

アナタは次のようなセキュリティ侵害に対してどのような対策を取っていますか? (ある意味、唯一の正解というのはない。)

□ 地震や火事やマシントラブルで重要なデータが失われる (マーフィーの法則: 起こってほしくない災難は必ず起こる)。

a. 計算機が壊れても平気なようにできるだけ使わない。
b. バックアップを取り、必要な情報が復元できるようにする。
c. お護りを購入する。

□ 他人があなたの名前を語ってイタズラメッセージを送る

a. メッセージに電子署名を施し他人に捏造できなくする。
b. 普段から人に信頼される態度を心がけイタズラであることを納得させる。
c. 24 時間、自分の名前がどこかに出て来ないか監視する。

□ 友人宛てのマル秘のメッセージを盗聴されてマル秘情報がばれてしまう

a. 本当にばれたら困ることはメールで送らない。
b. メッセージを暗号化して送る。
c. マル秘など持たない、表裏のない人間になる。

□ Web で通販購入時に盗聴されて、クレジットカード番号がばれてしまう

a. WWW の暗号化通信機能 (SSL) を使用しているサイトを選ぶ
b. 着払いや郵便振込で購入することにしてクレジットカード番号を打たない
c. 番号だけバレてもサインなしなら引き落とされないはずだから…

□ 自分が管理しているサーバに侵入されてしまう

a. 盗まれても困るデータは一切持たない

b. サーバをネットワークに接続しない

c. サーバとインターネットの間にファイアウォール (防火壁) 装置を入れておく

d. パスワードやユーザアカウントを厳重に管理する

e. 常時システムを監視し、ヘンな動作の兆候がないかチェック

これは結構面白いテーマ。まず、盗まれるものがないから侵入されてもいい、ということは言えない。侵入されると、そのあなたのサーバがよそへのいたずらの基地になってしまう。すると、あちこちから「あなたのサーバから侵入して来ようとしているからやめてくれ」という苦情がくる。だからデータの有無と関係なく、侵入されてはいけない。

その点で、一番安全なのはネットにつながないこと。実際、特殊用途のサーバではよく採用される手段。しかしそれでは Web サービスなどはできないから困る。

防火壁とは、特定のプロトコルだけ通過させる、特定のホストからの通信だけ許す、などさまざまな制約を設けて侵入者が自由にやるのを妨害するような「壁」。しかし壁があると正規の利用者にとっても邪魔になるので、あまり厳しくしすぎるのは考えもの (厳しくしすぎると利用者は抜け穴を作ろうとする)。

そういう意味では、地道だけどパスワード管理、システムの監視など管理者によるこまめな管理が一番重要。防火壁などを導入してもきちんと設定してなくて抜け穴があるなどでは無意味。

もう 1 つ、実は「セキュリティ侵害の 60%は内部者 (企業サイトなら社員) によるものである」という法則 (?) がある。もちろん、学校でも生徒がイタズラ、とういのは大いにありそうなシナリオ。規則の整備と教育が大切。

4 HTML 教室

4.1 再びスタイルシートについて

スタイルシートについては第 1 回でも説明したが、その後 HTML についてだいぶ復習したので、改めてここでスタイルシートとその利用についても復習する。基本的な考え方は次の通り。

- HTML では「段落」「見出し」「箇条書き」「箇条書きの項目」など、文書に出て来る「論理的な単位」を指定 (マークアップ) する。
- それぞれの単位をどのように「見せる」か (表現指定) は、スタイルシート機能で指定する。

このようにすることで、次のような利点がある。

- 「すべての見出しを青い文字に」など、統一したスタイルが指定できる。
- スタイルを変更したい場合にも、1 箇所だけ直せばすむ。
- HTML の属性として用意されていないような機能もスタイル指定では使える。(例: 要素ごとの背景色や背景画像指定、余白設定、枠の指定、など。)

4.2 CSS の書き方の復習

スタイルシートの書き方としては、HTML と組み合わせる場合、CSS(Cascading StyleSheet) と呼ばれる指定方法を使うのが普通。ここでもこれを扱う。これまで何回も見てきた

```
<style type="text/css">
body { background-color: white }
</style>
```

というのも CSS の指定。この<style>...</style>で囲まれた部分にいくらでも指定を書ける。
CSS の指定の基本的な形は次の通り。

セレクタ { プロパティ: 値; プロパティ: 値; … }

ここで「セレクタ」はとりあえず「タグ名」だと思ってもらえばよい。プロパティと値はそれぞれ「何を」「どのように」設定するかを指定。つまり「<h1>〜</h1>の、文字の色を、赤にして、文字飾りを、下線にする」だと

```
h1 { color: red; text-decoration: underline }
```

のようになる。

4.3 プロパティと値の指定方法

CSS での値の指定方法をまとめる:

- 長さ — 1mm(ミリメートル)、1cm(センチメートル)、1px(ピクセル…画面上の点の大きさ)、10%(外側— たとえばページ幅— に対する割合) など。
- 色 —「aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white, yellow」の 16 種類と、おなじみ「rgb(赤, 緑, 青)」(またはそれを 16 進数で表記した「#*rrggbb*」)。
- ファイル — 背景画像などファイルを指定する場合は「url(ファイル名)」のように指定。

CSS の代表的なプロパティをまとめる:

- color: 色 — 文字の色を指定する。
- background-color: 色 — 地の色を指定する。
- background-image: url(ファイル名) — 背景画像を指定する。
- font: 指定… — フォントの指定。大きさを表す名前またはパーセント値、フォント名などが指定できる…が、Unix 上の Netscape 4.x ではほとんど対応していない。
- text-indent: 長さ — 段落の 1 行目の字下げ量を指定。
- text-align: 指定 — left、right、center のいずれかで左揃え、右揃え、中央揃えを指定。
- text-decoration: 指定 — underline、overline、line-throught、blink、none が指定できる。
- margin-top:、margin-bottom:、margin-left:、margin-right: 長さ — 上下左右の余白を指定。
- padding-top:、padding-bottom:、padding-left:、padding-right: 長さ — 上下左右の詰めものを指定。余白と詰めものの違いは枠を引いてみると分かる。枠の外側が余白、内側が詰めもの。
- border: 指定 — 枠の指定。色、種類 (solid、double、groove、ridge、inset、outset、dashed、dotted、none のいずれか)、および幅を指定できる。
- float: 指定 — left または right を指定。これを指定した要素は左または右によせられ、空いた側にはテキストが流し込まれる。

4.4 演習: CSS のさまざまな設定を試す

今回も CSS の設定を行う演習をしていただく。

```
cp  ~kuno/work/css3.html  なんとか.html
```

でサンプルファイルをコピーしてきて、ブラウザで表示する。次にこのファイルに書き込まれている設問に従って CSS の指定を追加する。ファイルは次のようになっている。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="ja">
<head>
<title>Sample</title>
<meta http-equiv="Content-style-type" content="text/css">
<style type="text/css">
body { background-color: white }
div { margin: 1mm 2cm 1mm 2cm; border: green solid 2px }
/* この後ろに指定を追加して行くこと */

</style>
</head>
<body>

<h1>プログラム</h1>

<div>
設問 1: h1 の見出しを赤い文字、下線つきに。ヒント: 資料のすぐ上を見る
</div>

<p>
プログラムとは、コンピュータに対する命令を書き表したもので、次の
ような性質があります。
</p>

<div>
設問 2: p の段落で左マージン、右マージンとも 1cm に。ヒント: margin-left、
margin-right を設定する。
</div>

<ul>
  <li>決まった規則に従って書く</li>
  <li>先頭から順番に実行される</li>
  <li>繰り返しなどの制御構造を持つ</li>
</ul>

<div>
設問 3: ul の箇条書きで背景の色を黄色に。ヒント: background-color を設定。
</div>

<h2>簡単な例</h2>

<div>
設問 4: h2 の見出しに、青い (blue)、単一の (solid)、幅 4 ピクセル (4px) の枠をつ
ける。ヒント: border を設定する。
</div>

<p>ごく簡単な C 言語のプログラムを見てみましょう。</p>

<div>
設問 5: p の段落において、1 行目を 7mm 字下げする。ヒント: text-indent を設定。
```

```
</div>

<pre>
  int main() {
    printf("Hello, World.\n");
  }
</pre>

<div>
設問 6: pre の整形ずみテキストにおいて、左マージンと右マージンを窓の幅の
10%とし、文字の色を青 (blue) とし、背景色をピンク (pink) とし、全体を緑
(green) の、二重線の (double)、幅 6 ピクセル (6px) の枠で囲み、枠の内側を 10mm
あける。ヒント: margin-left、margin-right、color、background-color、
border、padding を設定。
</div>

<h2>プログラミングの難しさ</h2>

<div>
設問 7: h2 の見出しの文字の左右を 15mm あける。ヒント: padding-left、
padding-right を設定。
</div>

<p>
カーニハンという人は、プログラミングのコツについて次のようなこと
を言っています。
</p>

<div>
設問 8: p の段落の上と下を 5mm あける。ヒント: padding-top、padding-bottom
を設定。
</div>

<blockquote>
言いたいことを単純率直にいおう。わかりやすく書こう。「効率」のた
めに、わかりやすさを犠牲にしてはいけない。
</blockquote>

<div>
設問 9: blockquote の引用を紫 (purple) の文字、黄色 (yellow) の背景にする。ヒ
ント: color、background-color を設定。
</div>

<div>
設問 10: ここまでに設定した色をすべて自分のセンスに合わせて調整し直せ。ヒ
ント: rgb(赤, 緑, 青) 形式を使うとよい。
</div>
</body>
</html>
```