

計算機科学基礎'13 #5 – ネットワークサービスとセキュリティ

久野 靖*

2013.5.15

今回はネットワークの基本的な原理、つまりどのようにして世界中のどこでも情報をやり取りできるのかという話題を扱いました。今回はネットワークの話題の後半として、まず情報セキュリティについて学び、その後で今回は出てこなかった個々のネットワークサービスから代表的なものを取り上げて学びます。

1 情報セキュリティ

1.1 セキュリティとは/情報セキュリティとは

セキュリティ (security, 安全性) とはその名前の通り、安全が保たれるようにすること、たとえば身体的危険や財産の窃盗その他の危険にさらされないようにすることを言います。一般には、家屋にきちんと施錠したり、ガードマンに見回ってもらったり、財産なら安全な場所に預けるなどの対策があります。

では情報セキュリティはどうでしょうか。つまり「情報に関する安全性」ですが、それは一般のセキュリティとどう違うのでしょうか。情報社会の今日では「情報」自体に価値が置かれるようになっていますから、一般の財産と同じに考えればいいのでしょうか。たしかに違わない面もありますが、次のような点は違っています。

- コピーが容易であり、盗まれたかどうか分かりづらい。
- 同様に、書き換えも容易であり、変更されても分かりづらい。
- ネットワーク経由でのセキュリティ侵害が容易かつ一般的。

とくに最後の点は、今日の急速なネットワーク普及に社会が追い付いていない面があり、大きな問題です (なのでネットワークの回に取り上げているわけです)。

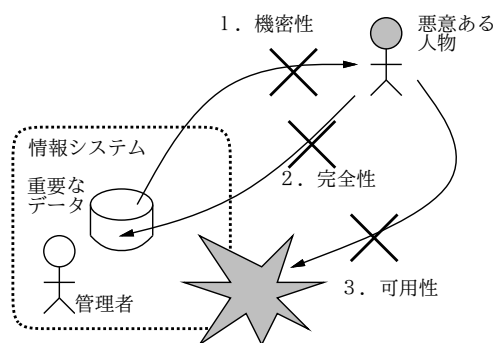


図 1: 情報セキュリティの 3 要素

では、情報セキュリティ対策を行うとして、その目標は何であるべきでしょうか？ 一般的には、次の 3 目標が挙げられます (図 1)。

*経営システム科学専攻

- 機密性 — 情報が、取得できるべきでない対象に漏洩しないこと。
- 完全性 — 情報が、完全な状態である、つまり改ざんされないこと。
- 可用性 — 情報が、必要なとき利用可能、つまりシステムがきちんと動作すること。

これらを見てみると、情報セキュリティ対策をきちんと行うことは、その多くの部分が人的な問題、管理上の問題だと分かります。たとえば機密性について言えば、ある情報が「誰は」見てもよくて「誰は」見ては困るか、という範囲が明確でなければ適正な管理のしようがありません。

完全性について言えば、ある情報をその権限を持つ人が業務の必要に応じて書き換えることは当たり前というかな必要なことですが、権限がない人が書き換えたり、権限がある人でも業務上の必然性がないのに不正に書き換えるのは問題です。実際、企業等におけるセキュリティ侵害の大多数は外部の人間ではなく内部の人間によるという調査結果があります。

可用性についても、いくらネットワーク経由の攻撃に対して防御していても、サーバ室に誰かが入ってきて電源を落としたり装置を破壊したらおしまいなわけです。地震や火事など自然災害への対策も重要です。なお、可用性に対する対策としては、システムを多重化してどちらかが壊れても運用を続行可能とする、重要なデータは定期的にバックアップして安全な場所に保管する、などがあります(このバックアップが流出すると機密性が損なわれるので注意)。

こうして見ると、組織の情報セキュリティ対策の責任を担うべきなのは、必ずしも情報システム部門だとは言えません。情報システム部門は技術的な対策には責任があるでしょうけれど、情報の取り扱い規定、機密性の分類、管理体制などはより上位の管理部門の仕事だと考えられます。

1.2 不正アクセス

不正アクセスとは、大まかに言うと(1)他人のID/パスワードを使ったり、(2)ソフトの欠陥などを突き通常のアクセス制限を回避して、システムを利用するような行為を言います。このような行為は、不正アクセス禁止法(1999 成立)によって禁止されています。

では、世の中にはどのような形の不正アクセスが多いのでしょうか? 特定の企業等から秘密を盗み出すため¹というものも急増していますが、「イタズラ」や「個人情報収集目的」のものもあります。したがって、自分のマシンに大したデータも入れていないので侵入されることはないだろう、と思うのは間違いなわけです。侵入されても自分のマシンには何も入れてないから問題ない? 侵入者はマシンを乗っ取ると、次にそのマシンを他者を攻撃するための「手下」(ボット)にします。ですから、侵入をほっておくと「お前のマシンがうちに攻撃が来ているぞ! 何とかしろ!」という警告を貰うかも知れません。

次に、そういう侵入を行うのは高度な技術が必要かということですが、パスワード推測やセキュリティホール²の活用によって「自動的に」他のサイトへの侵入を試みるプログラムが流通していて、これらを使うと自分では高度な技術を持っていなくても侵入が可能になります(そういうものを使う輩を Script Kiddie と呼びます)。

では、不正アクセスの被害に逢わないためにはどうすべきでしょうか? まず、本当に技術力のある侵入者に狙われた場合、完全に不正アクセスから逃れるのは難しいです。そのような場合は警察に相談するとともに、対策についてはその道の専門家をたのむべきでしょう。

上述の Script Kiddie などの「いたずら」に対しては、「あけっぱなし」「不注意」をなくし、侵入への「敷居を高く」することが有効です。あちらはどこでもいいのですから、ラクにイタズラできなければよそを試しに行くでしょう。そのような対策として、次のようなものがあります(図2)。²

- パスワード対策 — パスワードとユーザID をともに防御する(ユーザID の一覧も外部から隠す)。パスワードは推測されにくさのため、必ず記号などを含めるようにさせる。

¹ 「標的型攻撃メール」などがそうです。

² Web サービスなどの場合はさらに、サービス構築時に注意すべき点が多くありますが、ここでは触れません。

- セキュリティホール対策 — 使用している OS やアプリケーションのセキュリティ情報には気を配り、問題があればソフトを更新してそれを解消する。Windows なら Microsoft Update、Mac ならソフトウェアアップデートなどを動かし、重要度の高い修正は必ず適用する。
- 防火壁、パケットフィルタ — 外部ネットと内部ネットの間に防火壁 (ファイアウォール) を入れ、そこで「不要な」(提供するサービスやその管理に必須でない) ネットワーク接続を禁止する。

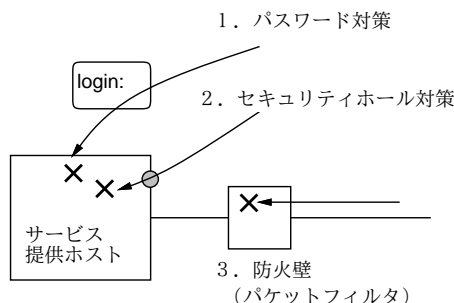


図 2: 代表的なセキュリティ対策

重要なのは、防火壁「さえ」あれば済むというわけではない、ということです。なぜなら、サービスの提供やその管理に必要な通信は防火壁を通過させる必要があり、これらの通信を経由した攻撃も必ず存在するからです (それ以前に防火壁をきちんと設定していないという論外な状態も実在するようですが)。さらに、内部者側の問題 (悪意ある内部者や、ウイルスへの感染など) にはまったく役に立ちません。

ですから、セキュリティのためには特定の 1 つの対策で十分ということではなく、さまざまな面から総合的に侵入への敷居を高くするべきだ、ということです。たとえば、図 2 では 3 つの攻撃が並列に書かれていますが、実際にはパスワードの不備や OS のセキュリティホールがあっても防火壁でこれらの不備を突く通信を遮断できれば被害を食い止められる、という関係があるわけです。

なお、敷居を高くすると、正規のユーザにとっても不便になることがあるため、ユーザが不満を持って「穴」をあけようとしたりすることがあります。このような問題については結局上で述べた「管理方針」の問題で、それがユーザにもきちんと納得されていなければうまく行かない、ということになるのではないのでしょうか。

1.3 マルウェア (俗にいうウイルス)

Winny を通じて広まるウイルスによる相次ぐ情報流出事件からも分かるように、コンピュータウイルス (以下単にウイルス) は今日の情報セキュリティ上の脅威として無視できないものとなっています。ではウイルスとは何でしょうか？

実は、ウイルスと言う場合、広義の意味と狭義の意味とがあります。広義のウイルスは狭義のウイルスとの区別のためマリシャスコード (Malicious Code、悪意のあるコード)、マルウェア (malware) と呼ばれることもあり、次のような性質を持つプログラム全般を言います。

- 何らかの手段で個人の PC などにとりつき、所有者の意図しないことを行うようなプログラム。

そして、広義のウイルスの中身はたとえば次のように分類できます (網羅的とは限りません)。

- 狭義のウイルス — プログラム、文書などのファイルの中に自分自身を埋め込み、そのファイルを開くことで他のファイルに感染し拡散するようなもの。
- ワーム — 単独のプログラムであり、ネットワークを通じて自分自身の複製を他のホストに送り込んで拡散しようとするもの。システムのセキュリティホールを突いて拡散するものと、メー

ルなどで自分自身を他のユーザ宛に送り込み、そのユーザをだまして自分自身を起動させようとするものがある。

- トロイの木馬 — 一見有用なプログラムのふりをして配布されており、起動するとそのように動作するが、その裏で悪事をはたらくようなもの。

悪事の内容としては、単に画面にイタズラを表示させる程度のものから、ファイルを勝手に削除する、マシンを起動できなくする、マシンに裏口(ウイルスを放った人がこっそり侵入できる手段)を設置する、利用者の打鍵をこっそり記録する、などがあります。

ワームでは他ホストへの感染のために大量のネットワークトラフィックを生成して迷惑となるものもありますし、勝手に大量のメールを他人あてに送るものもあります。この場合、感染したマシンのアドレス帳から宛先を取り出して送信するので、知合いに迷惑が掛かる(あなたもウイルスつきメールを受け取ったことがあると思います。しかも From:は別人に偽装されているので見た目の送信者に文句を言っても見当違いとなるわけです)。

近年とくに問題になっているのは、マシン内の情報を外部に送信するようなもので、多くの情報流出事件を引き起こしています。この流出手段も、Winny などのファイル交換機能によるもの、メールで送るもの、勝手に Web サーバになって送るものなどさまざまです。

では、ウイルスに対する防御はどうしたらよいのでしょうか。いわゆるウイルス対策ソフトは当然使用するのがよいですが、その基本的な動作原理は、ファイルが書き込まれる時などにその内容を調査して、パターンファイルに含まれる既知のウイルスのパターンと一致しないかどうか調べ、怪しければ書き込みを止めるなどです。

ということは、パターンファイルを新しいものに絶えず更新して行かないと、新たなウイルスには効果がないわけです。また、ファイルへの書き込みを行わないワーム(メモリ上でだけ自分の複製を作る)ではパターンファイルによる検出ができないという弱点もあります。

なお、Winny の漏出事件のときは、Winny が主に日本で使われているファイル交換ソフトだったためパターン更新が遅れたことが問題だったという説があります(さらに Winny の交換機能によるウイルスの拡散速度が速かったという面もあるでしょう)。

多くのウイルスはユーザが間違えて実行してしまうことにより感染しますから、もっとも基本的な防御手段は、要するに「未知のプログラムは実行しない」ということに尽きます。しかし、Winny で広まったウイルスの場合、自分自身のアイコンを「画像」「フォルダ」「テキスト」などの無害そうなもののアイコンにすり替え、名前もそれらしいものにしておくことでユーザが「うっかり」開いて(実行させて)しまうようになっていますから、十分な慎重さが必要なわけです。³

万一、ウイルスに感染した場合は、他のホストへの感染を防ぐためにネットワークから切り離し、駆除を試みますが、駆除できない場合はシステムを初期状態から入れ直すこととなります。このような場合のためにも、重要なデータは常にバックアップを取る必要があるわけです。

演習 5-1 以下の3つのテーマから1つ以上選び、過去に起こった「事件・事故」の事例を調べて、どのようなケースであり、何が原因であり、どのような被害があり、どうすれば防げたはずだったかを整理しなさい。できれば、3つ以上の事例を報告し、事例どうしの相互比較も行いなさい。

- パスワードの漏洩ないし安易なパスワードによる侵入やアカウントの乗っ取りの事例。
- 狭義のウイルスに感染した被害の事例。
- トロイの木馬によるマルウェア被害の事例。

「◎」の条件: (1)2つ以上のテーマについて、(2)「できれば」の部分までやってあり、(3)適切な(と担当が考える)整理・分析がなされていること。

³これから防御するには、すべてのファイルの拡張子をきちんと表示させること、長い名前で省略されているようならきちんとチェックすること、ネットワークから来たものを入れる場所はセキュリティレベルを上げて(Windowsなら「インターネット」ゾーンに設定)十分チェックさせること(こうするとプログラムの実行は常に警告が出る)、などがあります。

1.4 暗号技術とセキュリティ対策への応用

暗号技術は昔から、情報の機密性を保つために使われて来ました。そのことは、コンピュータ時代の今日でも変わりません。もちろん、計算機による高速な計算のため、使われる暗号方式はずっと高度なものになっています。しかし、暗号を破ろうとする側も同じ高速な計算が使えるわけですから、暗号を作る側と破ろうとする側のせめぎ合いは昔と変わっていないとも言えます。

どんな暗号であっても、十分な時間を掛ければ…すべての鍵の可能性をしらみ潰しに試すことで、破ることができます。ただしそれには、太陽が燃え尽きてなくなるまで掛かるだろうということで、実質破られないだろうとして扱うわけです。しかしこれには、計算機が高速になって試す時間が短くなってしまったり、また暗号方式に弱点があつて全部試さなくても部分的に試せば済んでしまう(従つて短時間で破られてしまう)可能性がある、という問題があります。

暗号には大きく分けて、次の2種類があります。

- 対称鍵暗号 — 暗号化と復号化に同じ鍵を用いる。
- 公開鍵暗号 — 2つの鍵の対を使い、片方で暗号化、他方で復号化を行う。

歴史的には対称鍵暗号の方が古くからありますが、これには「鍵をどうやって安全に伝達するか」という問題がつきまといまふ。ネットワークで送ればよいと思うかも知れませんが、送っているところで盗聴されたらその後の暗号化通信も解読されてしまいます。USBメモリなどに入れて会って手渡せば安全ですが、面倒すぎますね。

そこで考案されたのが公開鍵暗号で、こちらの場合は鍵を秘密鍵と公開鍵の対で生成し、公開鍵を皆に広く知らせてしまいます。暗号を使って通信したい人はこの公開鍵を使って通信を暗号化して送り、受け手は自分だけが持つ秘密鍵で解読します。他の人は秘密鍵を知らないのて、他人には解読できないわけです(図3)。

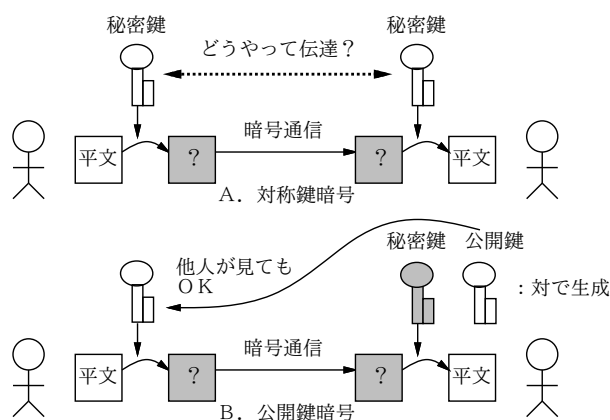


図 3: 対称鍵暗号と公開鍵暗号

また、公開鍵暗号の別の用途として、持ち主が持つ秘密鍵によってある情報に印をつけ、対応する公開鍵を用いてそれがOKであることを検証する(偽物の秘密鍵だとNGとなる)ことにより、情報を発信したのが確かに秘密鍵を持つ本人だということを証明する、というものもあります。こちらは書類にサインするのと似ていることから、電子署名 (digital signature) と言います。

ところで、公開鍵をネット経由で送ることに危険はないとしても、受け取った公開鍵が確かに本人のものかどうかを確認するという問題がまだ残っています。公開鍵は長い数値ですが、それをもとに生成したフィンガープリントという20~40桁くらいの数値があり、それを電話などで読み上げて照合することで間違いがないかどうかをチェックすることができます。しかしこれはこれで大変ですね…

そこでPKI (Public Key Infrastructure) という枠組みが用意されました。これは、CA (Certification Authority) という組織が他の組織や個人の公開鍵を「正しいと保証」するという制度で、保

証してもらいたい側は CA に自分の出自を示す書類を送るなどして確認してもらいます (もちろん、CA も商売なのでお金が掛かります)。では、CA 自体の正しさは…それは、その CA を保証する「親 CA」があり、そのまた親…とつながって行き、最後は「根元」の CA(ルート CA) にたどりつきます。ルート CA はそう沢山はないので、あらかじめ確認しておくなどの手段が取れます。なお、CA の仕事は「当該組織が存在していること」を確認し保証することまでであり、その組織の業務内容や社会的信頼性を調査して保証するわけではない点は注意しておいてください。

たとえば、ブラウザで暗号化されたデータ転送を行う SSL プロトコルでは、公開鍵暗号を使用し、この PKI を用いたチェックを組み込んでいます。そのため、代表的なルート CA について、ブラウザを配布する時に最初からその公開鍵情報がブラウザ内に組み込まれています。ブラウザで SSL のページ (https://で始まるページ) を開いた時は、このページの証明書情報からルート CA までの連鎖をたどれるかどうかをブラウザが自動的にチェックしています (図 4)。

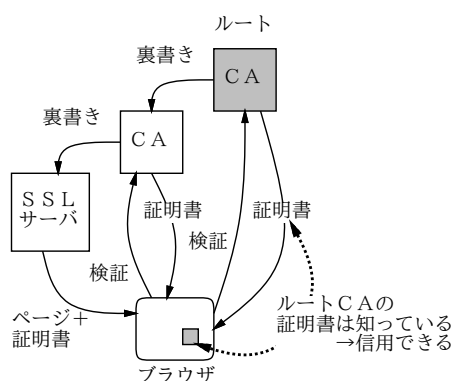


図 4: PKI と CA の連鎖

ところが、CA にお金を払うのが惜しいとか、海外の CA に行政機関が依存できないだろうとか (そうなんではなかね?) の理由で、このような連鎖に加わらずに SSL のページを動かしているサイトもあります。このようなサイトの証明書を (勝手に自分でいいだろと言っているというニュアンスで) 「オレオレ証明書」と呼びます。そのようなサイトを表示しようとする時、ブラウザは「ルート CA への連鎖がたどれません」という警告を出し、次の選択肢を提示します。

- (1) このサイトを恒久的に信用して今後警告なしで表示する。
- (2) とりあえず今回だけ信用して表示する。
- (3) 表示とりやめ。

このとき、基本は (3) ですが、どうしても見る用事があるなら「そのサイトの内容は信用できないかも知れない」という覚悟の上で (2) で見ることもありでしょう。決して (1) は選ばないようにしてください。

ところが、このようなサイトでは「SSL では暗号化はできているのだから、別に構わないので (1) を選んでください」とか「自分のところをルート CA に追加してください」とか呼び掛けていることが多いようです。いくら暗号化ができていても、見ているページそのものをどこが公開しているかの身元保証がないわけなので、決して従わないようにしましょう (行政機関などがこのようにせつかくの認証技術の価値を否定させるような行動を取るのも問題だと思われまます)。

最後に、普段のシステムへのログインなどパスワードに基づく個人認証技術も暗号技術が基本になっていることを注意しておきます (もちろん、指紋や静脈などの生体認証技術は別です)。

演習 5-2 ブラウザで適当な https: サイトを開き、鍵マークをクリックするなどの方法で「証明書情報」を表示させてみなさい。うまく表示できたら、以下のテーマから 1 つ以上 (できれば全部) やってみなさい。

- a. 自分が開いている `https:` サイトからルート CA までの CA の連鎖を確認してみなさい。できれば、複数のさまざまな国の `https:` サイトについて、連鎖がどうなっているかを整理してみられるとなおよいです。
- b. ブラウザの「証明書管理」画面を開き、そのブラウザにどのようなルート CA の証明書が標準で入れられているか見てみなさい。できれば、その中のいくつかについて、ネットで検索してどのような企業が調べてみるとなおよいでしょう。
- c. オレオレ証明書を使っているサイト (例: `https://kotonet.com/mail/inquiry1.html`) をブラウザで表示してみて、どのような警告が出るかを調べなさい。また、そのサイトの証明書も表示してみて、通常の (正しい) 証明書とどのような違いがあるか確認しなさい。できれば、GSSM の Mozilla には証明書が入っていないけれど Firefox には入っている `https://www.edinet-fsa.go.jp/EKWOAZ0015.html` などについても調べてみて報告できるとなおよいでしょう。

「◎」の条件: (1) 小問のうち2つ以上に解答しており、(2) いずれも「できれば」の部分までやってあって、(3) なおかつ適切な (と担当が判断する) 説明・考察がなされていること。

2 ネットワークサービス

2.1 ネットワークサービスとその構成

実際に私達がネットワークを使う時は、ただ単にパケットを送りたいわけではなく、メールを送るとか情報を調べるなど、何か「役に立つ機能」を使いたいわけです。ネット上でこのような機能を提供しているものをここではネットワークサービスと呼ぶことにします。

では、ネットワークサービスはどのような形で提供されているのでしょうか？ 前回見たように、ネット上で通信を行うためには、互いに通信する2つのプログラムが「あちら」と「こちら」で動いていなければなりません。「こちら」は自分のマシンだからいいとして、「あちら」はどうしたらいいでしょう？ この問題に対する回答は、プログラムを次の2種類に分けることです。

- サーバ — サービスを提供するマシンで常時稼働していて、サービス要求があるまで待ち、要求があったらサービスを提供する。
- クライアント — サーバに要求を出して、そのサービスを受ける。

このような方式をクライアントサーバ方式と呼びます (図5)。クライアントサーバ方式では、上述の通信の問題が自然な形で解決できますし、サービス提供のために必要な資源 (データ等) はサーバのところで一括管理できるので、各種サービスの実現が比較的簡単に行えます。このため、クライアントサーバ方式はネットワークの初期から今日に至るまでネットワークアプリケーションの構成方式として広く使われています。

クライアントは実際には1つではなく、そのサービスを利用するユーザの数に応じて多数動いているのが普通です。つまり、ユーザがネットワーク上のサービスを使おうとしたら、クライアントプログラムを起動し、このプログラムがサーバと接続してやりとりを行うわけです。言い替えれば、クライアントサーバ方式のシステムは、クライアントはそれぞれのユーザがいる多数のマシンで動き、サーバはサービスを提供するための専用マシンで1つだけ動く、という非対称的な構造を持ちます。

Unix では伝統的に、サーバのことをデーモン (daemon、守護神の意味) と呼びます。そしてネットワークサービスはとてども多数あるため、サービスごとにサーバを常時動かしていると待機プロセスだらけで負担が大きくなります。このため、Unix では `inetd` (Internet Daemon) と呼ばれるサーバがあり、専用サーバが動いていないサービスの要求を「代理で」待ち受けます。そして実際に要求が到着すると、`inetd` は「本物の」サーバを起動してサービスを行わせるわけです。もちろん、頻繁に使うサーバは `inetd` に頼まず常時動くようにします。

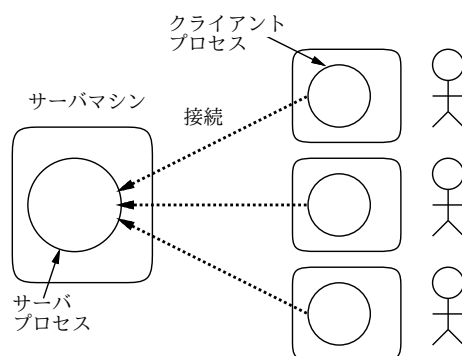


図 5: クライアントサーバ方式

ところで、クライアントサーバ方式ではない構成のネットワークアプリケーションもいくつかあります。これらはピアツーピア方式⁴と呼ばれ、特定のサーバはなく、各プログラムが対等な立場で通信します。このようなシステムで大規模なもの代表例としては、Napster、Winny、Skypeなどが挙げられます。これらはいずれも草の根的な通信やファイル転送を目的としているため、どこか1箇所に皆がアクセスするよりも、互いにやり取りをしたい組をうまく設定できさえすれば、それぞれが直接やり取りすることで負荷の集中を避けられるのです。⁵

では次節以降で、各種のネットワークアプリケーションとその構造について順次見ていくことにします。

2.2 遠隔ログイン

計算機ネットワークが作られたごく初期の時点から、「わざわざ遠くのマシンのところまで歩いて行かなくても手元のマシンの前に座ったままでそのマシンを利用できるようにしたい」という要望が存在し、それを可能にするソフトウェアが用意されていました。これを遠隔ログイン、ないし(手元のマシン上のソフトに遠隔端末の働きをさせることから)ネットワーク仮想端末と呼びます。そのためのコマンドとして今日の Unix では telnet、rlogin、ssh の3つが使われています。いずれも基本的には接続先ホストを指定して起動し、接続先ホストにおける自分のユーザ ID とパスワードを打ち込むとログイン認証が行われ、ログインすると接続先ホストのシェルが普通に使えます。

```
% telnet smb          ←ホスト指定して接続
Connected to smb.
Escape character is '^]'.
UNIX ...
login: kuno           ←ユーザ ID を入力
Password: *****    ←パスワード入力
Welcome ...
smb%                  ←接続完了
...                   ←普通に相手ホストで作業
smb% exit             ←終了
Connection closed ...
%                     ←元ホストに戻る
```

⁴筑波大学のネットワークではピアツーピア方式によるファイル交換ソフトの利用は禁止されています。

⁵中央サーバが無いので不法コピーなどが発見されにくいという点もあります。Winny は暗号化と組み合わせでこの部分売りものにしたソフトですが、その秘密主義がさまざまな不幸をもたらしてもいます。

歴史的には、telnet が最も古くから存在し、多くの機種で用意されています。続いて、Unix マシン相互でもっと簡単に使えるように、ユーザ ID を省略でき (手元のマシンを使っている時のユーザ ID を送る)、さらに特別な設定がなされていればパスワードも省略できる rlogin が作られ使われはじめました。⁶

遠隔端末ソフトでは、クライアント側はユーザがキーボードから打ち込んだ文字をそのまま相手ホストのサーバに送り、サーバ側ではそれらをシェル (ないしそれに対応するプログラム) に渡して実行させ、その出力をネットワーク経由で送り返してくるのでクライアントがそれを受け取って表示する、という形で動作しています。サーバ側ではログインした人の権限でコマンドを実行するため、ログイン認証が終わるとサーバ側で子プロセスを生成し、その子プロセスがユーザの権限でシェルを実行しています (図 6)。

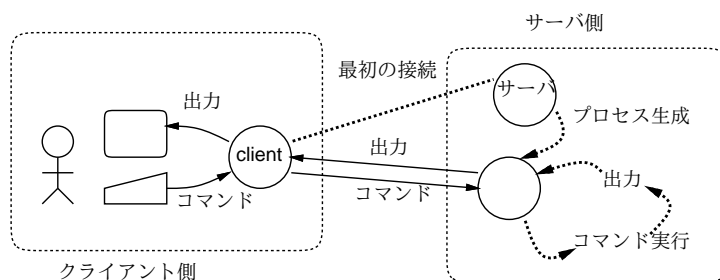


図 6: 遠隔ログインの原理

telnet⁷ と rlogin における問題は、これらのプロトコルでは往復する文字情報をそのままパケットに載せているので、(たとえばネットワーク診断用の機器やソフトを使って) パケットを傍受されるとパスワードも通信内容もすべて盗まれてしまうことです。

これでは余りに危険なので、通信路の両側で暗号化を行い、傍受があってもパスワードや通信内容が盗まれないようにしたのが SSH (Secure SHell) と呼ばれるソフト群と対応するプロトコルです。加えて SSH ではサーバ=クライアント間で公開鍵を利用して互いの正統性の確認も行います。

ssh はこのプロトコルを使って telnet 同様の遠隔ログインを行います。SSH は通常のパASSWORD 認証の他に公開鍵暗号を用いた認証もサポートしていて、こちらであればパスワードが盗まれる心配をさらに減らすことができ、より安全です (ただし、最初に公開鍵と秘密鍵の対を生成し、公開鍵を自分でログイン先に持って行って設置する必要があるため、設定は面倒です)。

SSH が提供するもう 1 つの機能として、ポート転送 (port forwarding) があります。これは、接続先ホスト内でしか使えないネットワークサービスを手元のマシンで使えるように接続を延長する「延長ケーブル」のようなものです (図 7)。たとえば、大学内の Web サーバには学内掲示が載っているため、そのサーバのコンテンツは外部には公開されていない、などの状況はよくあります。しかし大学まで行かなくても手元のマシンで学内掲示が見たいですね? そのとき、次のように「延長ケーブル」を使うことができます。

```
% ssh -L8080:smb:80 ユーザ名@接続先 ← smb のポート 80 を手元の 8080 に
password: *****
Welcome ...
>                                     ←接続完了
```

つまり、学内サーバのホスト名が「smb」だとして、接続先のマシンで smb のポート 80 番 (WWW サービスの標準ポート番号) に接続し、それを「延長」してきて手元のマシンのポート 8080 番として

⁶rlogin で接続先におけるユーザ名が手元のマシンでのユーザ名と違う場合は「-l ユーザ名」というオプションを指定することで相手先のユーザ名を指定できます。ssh では相手先の指定方法を「ユーザ名@相手先」という形にすることでユーザ名を指定します。

⁷最近の telnet では通信を暗号化する機能を提供しているものも一部あります。

取り出すわけです。そこで手元のマシンでブラウザを起動し、`http://localhost:8080` のページを開くと…学内サーバのページが読み出せるわけです。ここでは WWW を例に取りましたが、この方法で任意の TCP 接続を手元に延長して行うことができます。

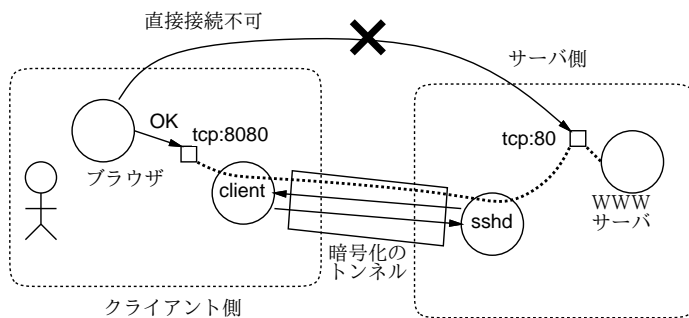


図 7: SSH によるポート転送

2.3 電子メールとネットニュース

電子メール (e-mail) とネットニュース (netnews) はネットワークの初期からの情報交換サービスで、インターネット以外に **USENET** と呼ばれる電話線や低速の通信回線により接続されたネットワーク上でも広く使われていました。⁸メールは今日でも広く使われていますが、ネットニュースについては日本では Web 上の掲示板などが主流になり利用が少なくなっています (しかし、米国などでは依然として活発に利用されています)。

メールとニュースは一緒になって発達してきたため、交換されるメッセージの形式なども良く似ていますし、システムの構造もよく似ています。すなわち、メールでもニュースでも遠隔地との情報交換は基本的に「サーバどうしで」行う設計であり、ユーザは「自分の手元の」サーバと通信してメッセージを投入したり取り出したりします (図 8)。

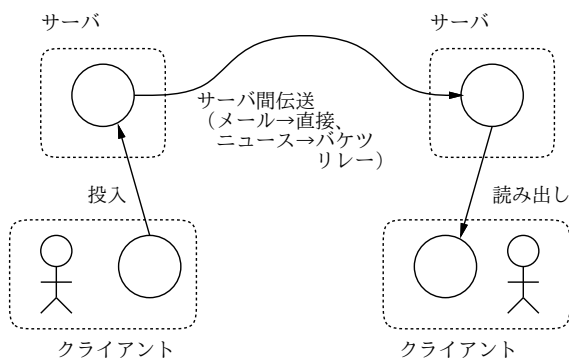


図 8: メール/ニュースの伝送経路

なぜそうなっているかという、インターネット以前のネットワークでは遠隔地との通信は時間がかかり複雑な制御を必要とする操作だったので、今日のように「ユーザがクライアントを起動して遠隔地のサーバと直接やり取りする」ことは困難だったからです。このため、そのような仕事は手元のサーバに依頼し、あとはサーバどうしでゆっくり情報を伝達してもらう、という構造になったわけです。なお、手元のサーバとのやりとりについても、古いシステムではサーバとユーザは同一のマシン

⁸日本では **JUNET** と呼ばれるネットワークが中心となって日本各地のサイトを結び、これらのサービスを相互運用していました。

上でファイルを共有することでメッセージを投入したり取り出していました。これはさすがに LAN の発達とともに少なくなり、ネット経由で手元のサーバにアクセスする構造に変わっています。⁹

なお、これらのシステムではサーバは普通にメールサーバ、ニュースサーバと呼ばれますが、ユーザがメッセージを読み書きするために使うクライアントは伝統的にメールリーダー、ニュースリーダーと呼ばれます。

電子メールの場合は、基本的にはユーザがメッセージを送信すると、そのメッセージは(メールリーダーから直接にせよ、手元のサーバ経由にせよ)宛先のサーバに届けられ、そこに格納されます。送信時のデータ伝達は **SMTP**(Simple Mail Transfer Protocol) と呼ばれるプロトコルで行われます。宛先のサーバはメールアドレスから分かるようになっています。たとえばメールアドレスは次のような形をしています。

someone@example.com

つまり「@」の後ろ側はドメインアドレス(この場合は example.com)になっています。ドメインアドレスは DNS を用いて IP アドレスに変換できることは既に説明しました。¹⁰IP アドレスが得られたら、そのアドレスの SMTP ポートに接続してそこに「someone さん宛ですよ」といってメッセージを送り込めばよいのです。

メールがメールサーバに到着したら、ユーザはそのメールサーバから自分のメッセージを取り出して読むわけです。このときユーザがサーバとやりとりするには、大きく分けて 2 通りの方法があります。

- ユーザが手元のマシンにメッセージを格納し管理する方法 — メッセージの投入は SMTP で行い、サーバから手元のマシンにメッセージを取り寄せる時には **POP**(Post Office Protocol) を使う。
- メッセージは基本的に常時サーバ内に格納しておき、ユーザはメッセージを読み書きするつどそのメッセージだけを手元のマシンとの間でやりとりする方法 — サーバと手元のメールリーダーの間での通信に **IMAP**(Internet Message Access Protocol) を使う。

前者ではメッセージが手元のマシンにあってその場で読んだり返信したりでき、サーバと通信するのはメッセージを送受信する短時間だけで済むという利点がありますが、あるマシンで読んでしまったメールはサーバから手元のマシンに移動してしまうので、後で別のマシンから同じメッセージを読もうとしてもそれはサーバにはなくなっています。ですから、マシンを 1 つだけ決めて常にそこでメールを読み書きするという使い方になり、場合によっては不便です。後者ではこのような問題がありませんが、その代わりにネットワーク経由でサーバとつながっていないとメールの読み書きができません。

ネットニュースの場合は、メールと異なり、個々のメッセージ(ニュース記事)には「宛先」はなく、ニュースグループと呼ばれる分類単位だけが指定されています。そこで、ユーザが手元のサーバにメッセージを投入すると、そのサーバは隣接するサーバに記事を転送し、そのサーバはさらに先のサーバに記事を転送し、というふうに「バケツリレー式」に記事が転送されていきます(逆に遠方からの記事も同様にして手元へ送られてきます)。このとき、記事の伝送の「堂々めぐり」が起きると大変なので、すべての記事には固有のメッセージ ID と呼ばれる識別名を割り当て、各サーバは手元に持っている記事のメッセージ ID 一覧を常に管理して、まだ持っていない記事だけを受け取ってリレーします。

ネットニュースではサーバに常時最新の記事群が大量に蓄積され相互に流通しているため、全部のメッセージを手元のマシンに取り寄せてから読むというのは非現実的で、読みたいメッセージだけを

⁹また、現在では遠隔地のメールサーバに直接アクセスしてメッセージを送信することもできます。ただ、向こう側のメールサーバが忙しかったりすると待たされるので、手元のサーバに投入してあとはサーバに任せる方が楽だし自然ということです。また、スパムメール防止のため、自ネットワーク外からの SMTP によるメール送信を禁止しているプロバイダも多々あります。

¹⁰ただし、メール伝送の時は DNS 上で「MX(Mail Exchange)」レコードと呼ばれるメール専用種別のデータを検索します。これは、メールアドレスの場合、あるアドレス(つまりメールサーバ)が停止していたら代替のサーバに送るなど、メール固有の扱いをする場合があるためです。

選んで個別に読むことになり、メールで IMAP を使うのと似た形になります。ただしプロトコルについては、サーバどうしの通信もサーバとユーザが使うクライアントの通信も同じ NNTP (NetNews Transfer Protocol) を使って行います。

メールでもニュースでも、伝達されるメッセージの冒頭部分にはメッセージヘッダと呼ばれる部分があり、ここに各種の管理情報が格納されています。その後ろにメッセージ本文がありますが、ヘッダと本文の間は 1 行の空白行 (長さが 0 の行) で区切られることになっています。ヘッダ部分の情報を見ると、そのメッセージがいつどこで投入され、どのように中継されてきたかが記録されていることが分かります。メールヘッダの一例を見てみましょう。

```
Return-Path: kuno@mail.ecc.u-tokyo.ac.jp
Delivery-Date: Tue, 16 Dec 2003 18:24:48 +0900
Return-Path: kuno@mail.ecc.u-tokyo.ac.jp
Return-Path: <kuno@mail.ecc.u-tokyo.ac.jp>
Delivered-To: kuno@gssm.otsuka.tsukuba.ac.jp
Received: (qmail 38329 invoked from network); 16 Dec 2003 09:24:48 -0000
X-qmail-remote-mx: 0
Received: from ns.ecc.u-tokyo.ac.jp (133.11.171.253)
  by utogwpl.gssm.otsuka.tsukuba.ac.jp with SMTP; 16 Dec 2003 09:24:48 -0000
Received: from m.ecc.u-tokyo.ac.jp (mail.ecc.u-tokyo.ac.jp [133.11.171.196])
  by ns.ecc.u-tokyo.ac.jp (Postfix) with ESMTTP id 5D84017EC82
  for <kuno@gssm.otsuka.tsukuba.ac.jp>; Tue, 16 Dec 2003 18:24:48 +0900 (JST)
Received: from mail.ecc.u-tokyo.ac.jp (as301.ecc.u-tokyo.ac.jp)
  by m.ecc.u-tokyo.ac.jp
  (Sun Internet Mail Server sims.3.5.2000.03.23.18.03.p10) with ESMTTP id
  <OHPZ00E4CE5BEV@m.ecc.u-tokyo.ac.jp> for kuno@gssm.otsuka.tsukuba.ac.jp;
  Tue, 16 Dec 2003 18:24:48 +0900 (JST)
Date: Tue, 16 Dec 2003 18:24:47 +0900
From: kuno@mail.ecc.u-tokyo.ac.jp
Subject: test
To: kuno@gssm.otsuka.tsukuba.ac.jp
Message-id: <OHPZ00E4DE5BEV@m.ecc.u-tokyo.ac.jp>
```

This is a test.

このように、ヘッダは「フィールド名: 値」の形の行が並んだもので、Subject: や From: などの情報もすべてヘッダフィールドとして伝達されています。ただし、誰が送信者かの情報はこれとは別に SMTP プロトコルでも伝達しています。この情報をエンベロープ **From** と呼び、Return-Path: ヘッダに格納されます。また、Received: ヘッダを見ると、メッセージがどのような経路を通過して中継されてきたかを追跡できます (途中のサーバがここに嘘を書き込んでいない限り)。

SMTP や NNTP などのプロトコルは、通常はメールリーダーやニュースリーダーがサーバとやり取りするために使うわけですが、通常の文字を使ったやりとりであり、簡単な構造をしているので、人間が手で打ち込んでみることもできます。たとえば、telnet クライアントでメールサーバに接続して自分あてに簡単なメッセージを送っている例を示しましょう。

```
% telnet utogw smtp ← SMTP ポートを指定してメールサーバに接続
Trying 10.1.1.1
Connected to utogw
Escape character is '^]'.
220 gssm.otsuka.tsukuba.ac.jp ESMTTP
MAIL FROM:<kuno> ←自分が誰かを示す (エンベロープ From)
250 ok
RCPT TO:<kuno> ←メール宛先を示すコマンド
250 ok
DATA ←「以下本文」コマンド
354 go ahead
From: kuno ←メールヘッダが最低 1 つは必要
←空っぽの行がヘッダ終わりを示す。
test... ←本文も 1 行以上あった方がよい。
. ←「.」だけの行があるとおしまいを表す。
250 ok 989909466 qp 19829
```

```
QUIT          ← 「これでおしまい」コマンド
221 gssm.otsuka.tsukuba.ac.jp
Connection closed by foreign host.
%
```

なお、ここで「嘘」を打ち込めばその嘘はそのまま相手に伝えられていくことに注意してください。メールヘッダの信頼性とはある意味その程度である、ということです。¹¹¹² ネットニュースの転送や読み書きに使われる NNTP もこれと同様の簡単なプロトコルです。

演習 5-3 携帯、フリーメール、会社のメールなどから GSSM のアドレスにメールを送ってみて (または友人に送ってもらってもよい)、そのメッセージヘッダについて次の点を 1 つ以上調べてみなさい。

- a. それらのメールがどのようなサーバを経て GSSM に到着しているかを調べる。できれば、3 つ以上の送り元のものを相互に比較し、違いを整理できるとなおよい。
- b. それらのメールが各サーバを経て来るのに、それぞれどれくらい時間を要しているかを調べる。できれば、3 つ以上の送り元のものを相互に比較し、違いを整理できるとなおよい。
- c. メールに写真その他のコンテンツを「添付」した場合にヘッダにどのような違いがあるかを調べる。できれば、複数種類のコンテンツについて試し、コンテンツの種類による違いを整理できるとなおよい。

「◎」の条件: (1) 小課題 2 つ以上に解答しており、(2) いずれも「できれば」の部分までやってあり、(3) 説明・考察が (担当から見ても) 適切な内容であること。

演習 5-4 utogw の SMTP サーバに telnet で接続して、自分あてに例題通りのテストメールを送ってみなさい。うまく行ったら、次の小課題から 1 つ以上 (できれば全部) やってみなさい。

- a. 自分あてのメールで、どういうことは「嘘」を書いてもそのまま通ってしまい、どういうことは「嘘」を書くと拒絶されるか、調べてみなさい。
- b. クラスメイト宛てにできるだけ「本ものらしい」メールを送ってみて、また通常のメールソフトでも同様のものを送ってみて、どういう違いがあるか検討しなさい。ただし日本語は使わないものとする。
- c. 自分あてでもクラスメイトあてでもよいので、「添付ファイル」のついたメールを送ることにチャレンジしてみなさい。添付するものの種別は何でもよいです。

「◎」の条件: (1) 小問に 2 つ以上解答しており、(2) それぞれに対する説明・考察が (担当の判断で) 適切であること。

3 World Wide Web

3.1 ハイパーテキストと WWW

今日、最も広く利用されているネットワークアプリケーションといえば電子メールと **WWW** (World Wide Web、Web) です。しかし、その見慣れた Web 画面の裏側がどのようにできているのかを考えてみたことのある人は、それほど多くないかも知れません。まずこの点から見ていきましょう。

Web の基本的な枠組みはハイパーテキストです。ハイパーテキストとは何か説明するとすれば、次のようなものになるでしょう。

¹¹しかし世界中のメールサーバを乗っ取ることは難しいでしょうから、メールヘッダを解析してやればどこまでが嘘でどこからが本当かはそれなりに分かるわけです。

¹²GSSM のサーバではローカルサイトからの送信ではエンベロープ From は実在するユーザかどうかチェックされます。ということは存在しないユーザは書けません。他人になります。すまません。

- 計算機の画面上にテキストや画像などの内容 (コンテンツ) が表示されている。
- コンテンツの中には、他のコンテンツやその特定箇所を「指し示して」いる箇所が埋め込まれている。これをリンクという。
- リンクの箇所を何らかの方法で選択すると、画面はそのリンク先の内容に切り替わる。
- このようなリンクで互いに結び合わされたコンテンツの集まりは、リンクを自由にたどりながら読み進んでいくことができる。

ハイパーテキストの概念そのものは WWW よりずっと前から存在しました。たとえば Macintosh で広く利用されてきたアプリケーションであるハイパーカードや、Windows のヘルプなどはリンクで結び合わされたコンテンツの集合体ですから、ハイパーテキストの典型例だといえます。Web も複数のページと呼ばれる単位がリンクによって結び合わされた構造を持っているので、ハイパーテキストの一種だといえます (図 9)。なお、Web をみるためのプログラムを一般に **Web ブラウザ** ないし単に **ブラウザ** と呼びます。¹³

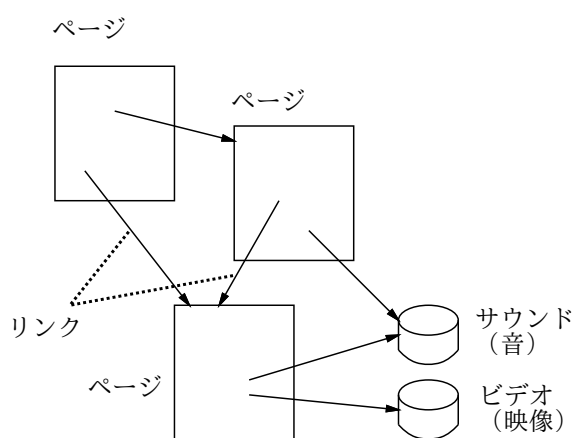


図 9: ハイパーテキストと WWW

ではハイパーテキストには普通の (紙の) 文書とくらべてどんな利点があると言えるでしょうか?

- 最初から順番に読まなくても、必要な箇所だけ選択して読むことができる。
- 計算機の機能を活用して、さまざまな支援ができる。
- 紙ではないので、画像や動画や音などのマルチメディアが取り入れられる。

最後の点に関連して、単なる文字 (テキスト) だけでないことを強調する場合には **ハイパーメディア** という言い方をすることもあります。

では、Web 独自の部分、つまりハイパーカードや Windows のヘルプにない部分というのは何でしょうか? それは次のような点だといえます。

- Web では、リンクは「ネットワーク上の任意のページ」を指すことができ、従って世界中の情報源を行き先とすることができる (World Wide の意味。ちなみに Web は「くもの巣」の意味で、リンクが網目状にはりめぐらされていることを表しています)。
- さらに、リンクはページだけでなく、「ネットワーク上のさまざまなモノ (資源)」を指すことができる。

¹³本来ブラウザというのは「色々なものを眺め回るためのツール」という意味であり、Web とは関係ないブラウザもあるのですが、Web があまりにも普及したため、ブラウザと言えば Web ブラウザの意味で通用するようになってしまいました。

- それらの資源の中には、単なる受動的なデータではなく、自らが能動的に動くようなもの、さらには「読み手」から情報を受け取るものも含まれている。
- これらの情報源はリンクをたどった瞬間にネットワーク経由で取り寄せられるので、ディスクや CD-ROM の容量などの制限とは無縁であり、また常に「発信されている最新の情報」が取り出せる。
- Unix、Macintosh、Windows という主流のプラットフォームすべてにブラウザが用意されており、どのプラットフォームでもどこから来た情報でも表示できる (クロスプラットフォーム)。

1990 年に WWW をはじめて作り出したのは、当時スイスの **CERN**(欧州粒子物理研究所) に所属していた Tim Berners-Lee です。彼の発明以前からインターネットはずっと存在し、電子メール、ネットニュース、FTP (ファイル転送サービス) などの手段で情報を流通することは可能だったのですが、これらは操作方法もバラバラで、どこに何があるか熟知していなければ情報にアクセスすることができませんでした。ここにハイパーテキストの考え方を取り込み、情報のありかや種別を知らなくてもページに内包されたリンクを選択するだけでその情報にアクセスできるようにした、というのが WWW の偉大な発明だったわけです。

CERN で作られた最初の Web ブラウザはテキストしか扱うことができませんでした。NCSA(米国アリゾナ州のスーパーコンピュータ応用センター) の学生たちが多くの Unix システムで動き、ページ中に画像を含められるブラウザ **Mosaic** を開発し公開したことで、Web は急速に普及し始めました。その後、Mosaic の開発者たちは Netscape Communications 社に移り、より高度で Unix 以外に Windows や Macintosh でも動く **Netscape** ブラウザを発売し、巨額の利益をあげました。しかし他の企業にインターネットという巨大市場を取られまいとした Microsoft 社が Windows にブラウザ **Internet Explorer** を無償で付属させる戦略を取ったため、Netscape 社はブラウザで儲けることができなくなり没落しました。¹⁴

一方、WWW のさまざまな規約 (プロトコルや記述言語) の標準に対するニーズも急激に高まり、これを受けて **WWW コンソーシアム (W3C)** と呼ばれる共同体が発足し、標準の取りまとめや参照用のソフト開発を行うようになりました (Tim Berners-Lee も W3C の発足とともにここに移っています)。歴史の話は詳しくするときりがないのでこれくらいにして、先に挙げた WWW の特徴が実際にどのような形で現われているかを、もう少し見て行くことにしましょう。

3.2 URL とリンク

WWW では「さまざまなモノ (資源、resource) を指すポインタ」として **URL**(Uniform Resource Locator) と呼ばれる形式を使用しています。URL は一般に次の形をとります。

スキーム:アドレス

そしてその中で最もよく使われるのが、Web サーバからコンテンツを取り寄せるためのプロトコルである **HTTP** (HyperText Transfer Protocol) を使う場合で、次のように指定します。

`http://ホスト指定/ディレクトリ/.../ディレクトリ/ファイル`

なお、サーバとブラウザの間で暗号化通信を行うために HTTP に暗号を入れた **HTTPS** プロトコルもあり、その場合は上の `http:` を `https:` に取り換えます。

なぜここで URL について長々と説明しているのかというと、URL こそが WWW の「肝」だからです。後で見るように、WWW でコンテンツを伝送しているのはごく簡単なプロトコル (実質はファイル転送のようなもの) であり、ブラウザがやっていることは基本的には次の 2 点だけです。

¹⁴さらに Microsoft は、パソコン製造各社に Netscape を搭載しないよう働きかけるなど公正でない活動をしたとして訴えられたりもしています。

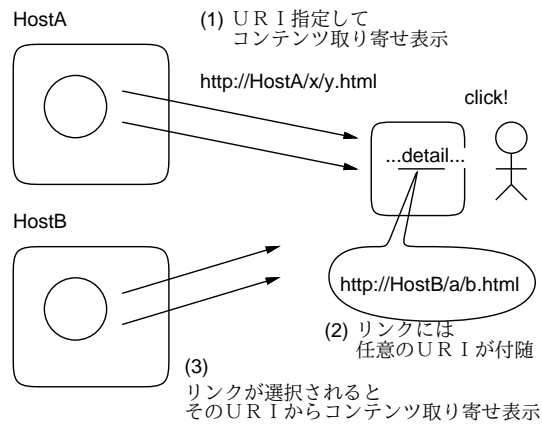


図 10: リンクの仕組み

- a. 指定された URL からコンテンツを取り寄せる。
- b. 取り寄せたコンテンツを適切な形で画面に表示する。

ここでコンテンツを取り寄せたとき、それが **HTML** 形式ファイルであれば、中にリンクが含まれていることがあります (そこに別のコンテンツを指す URL が埋め込まれています)。そしてユーザがそのリンクを選択すると、ブラウザはそのリンク先のコンテンツを取り寄せて表示画面をそちらに切り替えます (つまり上記 a の動作を行います)。これで結果的に「別のページへ飛ぶ」わけです (図 10)。これだけで済むのは、「リンク先」が URL の形で表されていて、世界中のどのサーバのどのコンテンツであっても自由に指し示すことができるから、なのです。

3.3 HTML の初歩

ではここで、ごく簡単に HTML とはどんなものか説明し、自分のページを作って見ましょう。GSSM では各自のホームディレクトリの下に `www` という名前のサブディレクトリを作ってそこに HTML ファイルを入れるだけで、内部サーバ経由にページを公開することができます。以下の作業は最初の 1 回だけ行ってください。

```
% mkdir WWW
% chmod a+rx WWW
% touch WWW/index.html
% chmod a+r WWW/index.html
```

あとはこの `WWW/index.html` を Emacs などで編集するだけです。とりあえず、次のような内容を入れてみてください (○○のところは自分の名前をどうぞ)。

```
<!DOCTYPE html>
<html>
<head>
<title>○○'s page</title>
</head>
<body>
<h1>○○です。 </h1>
<p>…挨拶ないし自己紹介を書く…</p>
</body>
</html>
```


これを作成した後、ブラウザで `http://w3in/~ユーザ/` を表示してみてください (図 11)。



図 11: HTML をブラウザで表示

または、GSSM 内のファイルに直接書き込めない状況で練習するために、計算機科学基礎 2013 のページに「HTML 練習環境」も用意しました。こちらはページ上部の欄に HTML を打ち込んで (全部打ち込まなくても大枠はコピー用のものがページの下部に用意してあります) 「Run」ボタンを押すとすぐ下の部分にその表示が現れます (図 12)。

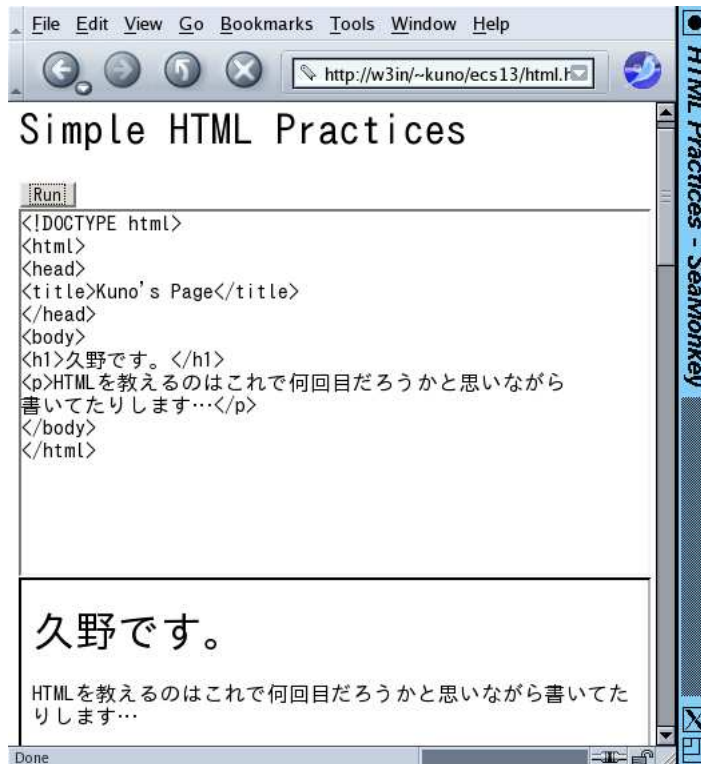


図 12: HTML をブラウザで表示

冒頭の「`<!DOCTYPE html>`」というのは以下のファイルが HTML であることを宣言します。以後、HTML では `<名前>…</名前>` という形でさまざまな種類の範囲を指定します。これを要素、最初の `<名前>` は開始タグ、最後の `</名前>` を終了タグと言います。つまり開始タグから終了タグまで 1 つの要素が構成されるわけです。要素の中には別の要素が入られます (入れ子構造)。上に出て来る HTML の要素 (タグ) の意味は次の通り。

- `<html>…</html>` — HTML 文書全体をあらわす。
- `<head>…</head>` — ヘッダ (この文書に関する情報を記述する部分) をあらわす。

- `<title>…</title>` — 文書のタイトルをあらわす。
- `<body>…</body>` — 文書本体 (ブラウザの窓の内側に表示される内容全体) をあらわす。
- `<h1>…</h1>` — レベル 1 の見出し (大見出し) をあらわす。さらに小さいレベルの見出しとして `<h2>…</h2>` ~ `<h6>…</h6>` まで使うことができる。
- `<p>…</p>` — 通常の段落をあらわす。

なお、段落等の中はすべて窓の幅に合わせて詰め合わせられますから、空白や改行を入れても意味がありません。空白や改行で整形したい場合は `pre` 要素を使います。

- `<pre>…</pre>` — この内側は空白や改行をそのまま残す。

次のような形のを末尾 (ただし `</body>` の前) に入れてみると図 13 のようになります。

```
<pre>
柿食へば
    鐘が鳴るなり
        法隆寺
</pre>
```



図 13: `pre` 要素は空白と改行が残る

最後にリンクのつけ方をやってみましょう (`h2` 要素はおまけ)。

- `リンクテキスト` — 「リンクテキスト」の部分がリンクとして表示され、そこを選択すると `URI` のページを表示する。
- `<h2>…</h2>` — 第 2 レベルの見出し (中見出し) を表す。

このように、`a` 要素の開始タグは「`href=...`」という指定 (属性と呼びます) がつけられ、それによってリンク先を指定します。これを利用して、「リンク集」を作ってみましょう。1 項目を 1 段落とします (リンクは段落等の中に入れることになっています)。例を示しましょう (図 14)。

```
<h2>私のリンク集</h2>
<p><a href="http://www.fujitv.co.jp/">フジテレビ</a>のサイト
はよく見に行きます。ドラマ好き。</p>
<p><a href="http://www.shes.net/dramacheck/">あらすじネタバレ</a>
なんていうのも愛用しています。</p>
```

なお、ここまでの説明だと WWW は HTML を取り寄せて来て表示するだけのシステムということになりますが、実際にはもっと対話的に、ブラウザ内に入力してそれをサーバに送って処理するなどのことも行えます (そういうページを誰でもよく使っているはずです)。そのあたりは「計算機ソフトウェア」で扱います。



図 14: リンクを入れたページ

演習 5-5 専攻内サーバに自分の Web ページを作成しなさい。表示ができたなら、次のテーマから 1 つ以上 (できれば全部)、作成/検討してみなさい。

- 見出しのタグは<h1>~</h1>から<h6>~</h6>まで 6 レベル用意されていますが、それらがどのように見た目を違えているか (大見出しはより大見出しらしいか) 調べてみなさい。できれば、複数のブラウザでやってみて比較できるとなおよいです。
- 段落のタグ<p>~</p>はどのような機能を持っているか (たとえば字下げとか複数並べたときの様子とか)、検討してみなさい。できれば、ディビジョンのタグ<div>~</div>と引用範囲のタグ<blockquote>~</blockquote>についても同様に調べて、相互に比較できるとなおよいです。
- 自分のページ内にリンクを作ってみなさい。たとえば<pre>~</pre>の内部にもリンクが作れるかやってみなさい。さらに、リンク集でなく自己紹介などの普通の文章の中にリンクが埋めてあるものも作ってみられるとなおよいでしょう。

「◎」の条件: (1) 小課題 2 つ以上について、(2) 「できれば」の部分までやってあり、(3) やって見た結果分かったことについて (担当者の考える) 適切な記述がなされていること。

3.4 WWW サーバとブラウザのやりとり (HTTP)

最後に、WWW のしめくりとして、Web サーバとブラウザがどうやって通信しているのを見てください。SMTP と同様、HTTP もごく簡単なプロトコルで、telnet コマンドを使って直接打ち込んで試すことができます。

```
% telnet w3in http      ← HTTP プロトコルで接続
Trying 10.1.0.3...
Connected to smb.      ←サーバに接続された
Escape character is '^]'.
GET /~kuno/           ← GET で URI のサーバ名より後ろの部分指定
<!DOCTYPE html>      ←ここから返送内容
<html>
<head>
<title>Kuno's Page</title>
</head>
<body>
<h1>久野です。</h1>
<p>HTML を教えるのはこれで何回目だろうかと思いながら
書いてたりします…</p>
<pre>
  柿食へば
    鐘が鳴るなり
      法隆寺
```

```
</pre>
```

```
<h2>私のリンク集</h2>
```

```
<p><a href="http://www.fujitv.co.jp/">フジテレビ</a>のサイト  
はよく見に行きます。ドラマ好き。</p>
```

```
<p><a href="http://www.shes.net/dramacheck/">あらすじネタバレ</a>  
なんていうのも愛用しています。</p>
```

```
</body>
```

```
</html>
```

←ここまで返送内容

Connection closed by foreign host. ←おしまいになる

つまり、サーバにコンテンツのパスを打ち込むと直ちにその内容が返送され、接続が切断されます。このように簡単にしておくことで、ブラウザはあちこちのサーバのページをロードし、サーバは現在どこのブラウザが自分につながっているかをいちいち管理しなくても済むため高速なサービスが行え、大量のクライアントを扱えるようになるのです。¹⁵

演習 5-6 自分の作成したページについて、telnet を使って取り寄せてみて、自分の作成したソースと同一であることを確認しなさい。うまく行ったら、普段よく見ているページを1つ選び、そのページをtelnetでアクセスしてみて普段と同じように役立てられるか検討しなさい。できれば、その中に含まれているリンクを辿ってみられるとなおよいでしょう。

「◎」の条件: (1) 「できれば」の部分までやってあり、(2) 適切な(と担当が考える)結果が得られ、説明がなされていること。

4 まとめ

今回は、実際にネットワーク上で私達がお世話になるようなネットワークアプリケーションとその仕組みについて、代表的なものを選んで概観してきました。これらをざっと見ただけでも、ネットワークには新しいものが生まれて来る余地が無限にあることが分かると思います。

¹⁵ 毎回 TCP 接続を張り直すのは無駄が大きいのので、1つ取り寄せた後も次の内容を取り寄せるために接続を切らずに応答を続けるような指定も可能になっています。