百萬石 2006年 春号
MMA[1]

目 次

[1] みんなで マイコンで 遊ぼう会の略と Microcomputer Making Association の略の 2 つの説がある.

はじめに

新入生の皆様、ご入学おめでとうございます。2006 年度部長の oku です。

このページでは MMA について簡単に解説します。

MMA は「Microcomputer Making Association」とか「もっとマイコンで遊ぼう」の略だと噂されています。
拡張子風に「えむえむえー」と発音してくれれば、電通大生の 20% くらいは「あぁ、あのよく分からないサークルか」といった反応をしてくれるものと思います。

活動している自分たちも「MMA とは何たるか」よりは、やりたいことをやりたいようにやっているので、
工学研究部のような類似のサークルに比べると分子間力が弱い気もします。

平たく言えば「コンピュータに興味のある人の集まり」です。
MMA では、基本的に PIC とか H8 とか 8051 のようなマイクロコントローラから普通の PC 位までのコンピュータに関わる活動を主にやっています。

というわけで、コンピュータに興味のある人はどうぞ覗いてみてください。サークル棟 2 階奥です。

第 I 部
脅威と研究の新地平

(著者) moechar
moechar@mma.club.uec.ac.jp

概要
新入生のみなさん，御入学おめでとうごめでとうございます．本稿は昔流行ったウイルスなどを引き合いにだして最近たびたび報じられる情報流出に備えたり，本学の学部の講義では教えてもらえない神経回路学の知識を持ちだして他学生より一歩先を歩もうとかいった優等生的な趣旨に基づいているわけではなく，単に今となっ ては懐かしいものを振り返ってみたり，我々が組織化して以来頭部に有する脳の仕組みを人工的に模倣できないだろうかという極めてポジティブな issue の元に記述されました．[2]

1 Code Red

2001 年あたりに Code Red なるウイルスが流行っていたのでこのウイルスの挙動を振り返ってみましょう，また，Code Red には亜種が存在しますが本稿ではその内の Code Red 2 について取り扱いました．
Code Red は Web サーバーである Internet Information Server[Services] のセキュリティホールを利用して不正な活動を行いました．このセキュリティホールとは

> Index Server ISAPI エクステンションの未チェックのバッファにより Web サーバーが攻撃される (MS01-033)

というものであり，起こりうる被害として，

> 悪意あるコードを含んだ HTTP リクエストを送ることによりバッファオーバーフローを発生させてこの Code Red に感染する

というウイルスでした．
Code Red は感染した計算機で下記のような挙動をとりました．

- IIS の異常終了
- 広域スキャンによるネットワーク速度の低下
- コンテンツの改竄 (英語版でのみ発生した)
- トロイの木馬の設置
- 他の IIS サーバーを探して伝染する

Code Red は Windows NT/2000 に感染し，特に 2000 の方での被害が尋常ではありませんでした．上記の挙動について詳しく見てみました．

(1) バックドアのインストール
IIS のスクリプト用フォルダに cmd.exe を C,D ドライブの特定のフォルダに root.exe という名前でコピーする．

(2) トロイの木馬のインストール
C,D ドライブのいちばん上の階層に窓使いにはおなじみの explorer.exe を呼び出すプログラムである explorer.exe を作成する．この explorer.exe が次項のような挙動を取りました．

(3) レジストリの書き換え
explorer.exe の呼び出しによりレジストリキー[3]

HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon

[2] 1 節と 2 節の内容的な初出は 2004 年秋号です．悪しからず．
[3] OS の仕様で円印がバックスラッシュになっています．

のキー SFCDisable に値 0xFFFFFF9D がセットされました. これによって Windows のシステムプロテクション機能が無効化され重要なファイルの変更が可能となってしまいました.

(4) 仮想ディレクトリの作成
explorer.exe の働きによって読み, 書き, 実行が可能となった仮想ディレクトリ C,D が作成されました. これらの中身にはドライブ C,D が Mapping されていました.

(5) 他のホストへの感染活動
感染後, ほかのホストに対して感染活動を開始しました. 感染活動用スレッドを 300 個作成し, それぞれのスレッドで 24 時間の間感染活動を行いました. 活動後にシステムを reboot し, 最終的に Code Red がメモリから消えてバックドアの root.exe と explorer.exe が残りました. この二つの実行ファイルが存在しているかどうか周りの人間が検索にかけて処置に困っていたのが記憶に残っています.

Code Red に感染して最も困ることといえば自分の意思と関係なく他のサーバーを攻撃してしまうことと, ホストアドレスが漏洩した場合に他人から内部ファイルを操作されてしまうということが挙げられます. 後者についての方法論は法律に触れるないしは種々の罪に問われる恐れがあるため提示しませんが, 当時の中流クラッカー達は楽しげに侵入していたようです. 今日でも晒されている人を見付けたらただ外から顔文字付きでかわいそすと同情するだけでなく[4], ちゃんと教えてあげましょう.

2 SirCam

SirCam も Code Red と同じ様な時期に蔓延したウイルスでした. Code Red が NT 系の OS に感染したのに対して SirCam は 95/98/98SE/Me に感染しました. 感染媒体はメールの添付文書ファイルであり, バイナリ切断を行うことで感染ファイルから文書部分のみを切り離すことも出来ました. SirCam に感染するとそれは下記のような挙動をとりました. [1]

(1) %TEMP%\<ファイル名>と C:\Recycled\<ファイル名>として

(2) 自分自身を C:\Recycled\Sirc32.exe と%System%\Scam32.exe の名で複製しました.

(3) 値 Driver32=%System%\scam32.exe をレジストリキー

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices

に追加しました.

(4) HKEY_LOCAL_MACHINE\Software\SirCam というレジストリキーを作成し, 下記の値をセットしました.

- FB1B - ごみ箱に保存された SirCam のファイル名を保存
- FB1BA - SMTP IP アドレスを保存
- FB1BB - 送信者のメールアドレスを保存
- FC0 - SirCam の実行回数を保存
- FC1 - SirCam のバージョン番号を保存
- FD1 - 実行された SirCam の拡張子を除いたファイル名を保存
- FD3 - SirCam の現在の状態に関する値を保存
- FD7 - この処理の実行前に送信されていたメールの数を保存

(5) HKEY_CLASSES_ROOT\exefile\shell\open\command というレジストリキーの値が C:\recycled\sirc32.exe "%1" %*" に設定されました. これにより .exe ファイルを実行する度にワームが実行されるという厄介なフラグがたったのです.

[4] 勿論, 複数回同情して王様化したりジェットストリームの勢いがついてもだめです.

(6) ネットワークを通じて共有されている他のシステムを探し， 発見すると，次の挙動をとりました.

- <コンピュータ名>\Recycled\Sirc32.exe に自分自身をコピーする.
- <コンピュータ名>\Autoexec.bat ファイルに"@win \recycled\sirc32.exe"の行を追加する.
- <コンピュータ名>\Windows\Rundll32.exe を <コンピュータ名>\Windows\Run32.exe にコピーする.
- <コンピュータ名>\Windows\rundll32.exe を C:\Recycled\Sirc32.exe で置き換える.

(7) $\frac{1}{33}$ の確率で自分自身を C:\Recycled\Sirc32.exe から %Windows%\Scmx32.exe に複写し，さらにレジストリキー

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\ShellFolders\Startup

が示す場所に自分自身を"Microsoft Internet Office.exe"として複写しました.

(8) 毎年 10 月 16 日になると，$\frac{1}{20}$ の確率で C ドライブのすべてのファイルとフォルダを削除する. この発病症状は日付形式が D/M/Y（日/月/年）に設定されているコンピュータ上でのみ発生する. さらに， SirCam に含まれるファイルに FA2 という文字が連続して含まれ，そのすぐ後に sc という文字が続いていない場合， 日付や日付形式に関係なく，すぐに発病処理が作動するという能力を持っていました. しかし，乱数値生成コードの初期化部分にバグがあったため，実際にはファイル削除およびスペース埋め尽くしの発病症状が発症する可能性はほとんど無かったと言って良かったでしょう.

(9) 上記の症状が発症した場合， C:\Recycled\Sircam.sys を作成し， ディスクスペースがなくなるまで文が追加されました. 文は次の 2 つのうちのいずれかです.

```
[SirCam_2rp_Ein_NoC_Rma_CuiTzeO_MicH_MeX]
[SirCam Version 1.0 Copyright 2000 2rP Made in / Hecho en - Cuitzeo, Michoacan Mexico]
```

(10) SirCam はメール処理に使う独自の SMTP エンジンを持っていて， 次の 2 通りの方法でメールアドレスを取得しました.

- 次の 2 つのレジストリキーで参照されているフォルダから sho*.，
  get*.，hot*.，*.htm ファイルを検索する.

  HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Cache
  HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Personal

  そして， 見つかったファイルに含まれるメールアドレスを%Windows%\sc?1.dll ファイルにコピーする.
  ? の部分は， アドレスを取得した場所によって次のように異なる.

  - scy1.dll: %cache%\sho*.，hot*.，get*. の場合
  - sch1.dll: %personal%\sho*.，hot*.，get*. の場合
  - sci1.dll: %cache%*.htm の場合
  - sct1.dll: %personal%*.htm の場合
- %system%とすべてのサブフォルダにある*.wab（Windows Address Book すべて）を検索し，
  そこに登録されているメールアドレスを%system%\scw1.dll ファイルにコピーする.

(11) 次のレジストリキーで参照されているフォルダから， 拡張子が.doc， .xls， .zip のファイルを探し，

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Personal

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Desktop

発見したファイルのファイル名を%system%\scd.dll に保存する．これらのうち，
いずれか 1 ファイルが SirCam のオリジナルの実行可能プログラムに付加され，
それがメールの添付ファイルとして使用されることになります．
また， 差出人欄のメールアドレスとメールサーバはレジストリキーから取得しました．．
メールアカウントが存在しない場合， 現在のユーザ名に"prodigy.net.mx"を付加しました．例えば， JSmith としてログオンしている場合， SirCam によって使用されるメールアドレスは，
jsmith@prodigy.net.mx となる．その後， SirCam はレジストリから取得したメールサーバか，
次のいずれかのメールサーバに接続を試みました．
prodigy.net.mx
goeke.net
enlace.net
dobleclick.com.mx

3 ニューラルネット序論

本節ではニューラルネット研究の歴史的変遷を踏まえつつその 2，3 のキーワードについて紹介します．[5]

3.1 ニューロンモデル

生物学上の**ニューロン**とは人間を含めた動物の脳を構成するもっとも基本的な素子です．電子顕微鏡の開発よってニューロン同士が不連続な構造であることが証明されました．数字を示せば，人間を脳は大脳と小脳をあわせて約 10^{11} 個ものニューロンが繋がっててきた壮大なネットワークなのです．ニューロンの生物学的な性質が明らかになるにつれてそれを単純化し数理的な挙動をモデル化した**ニューロンモデル**が数多く提案されてきました．そのうち，McCulloch と Pitts が提案した McCulloch-Pitts ニューロンモデル (1943 年) について説明します．[3, p.2-3]

あるニューロンに対して他のニューロンからの入力 x_k が通過する結線にはシナプス結合の強さを表す**重み** w_k が付けられ，その重み付き和 $\sum w_k x_k$ がしきい値 θ を越えたときニューロンが**発火** (1 を出力) し，越えないときには発火しません (0 を出力します)．重みの値域は実数集合 $\mathbb{R}$ で定義されるため，負の値もとり得えます．このことから，正の重みは発火を促す**興奮性**の，負の重みは発火を抑える**抑制性**の働きをすることがわかります．このとき，ニューロンの出力関数 f は以下のように書けます．ただし，$1(h)$ は Heaviside 関数であり $h \geq 0$ のとき 1 を，$h < 0$ のときは 0 を返します．

$$f = 1(h), \qquad h = \sum_{k=1}^{K} w_k x_k - \theta \tag{1}$$

K はニューロンへの入力結線の数です．

3.2 ニューラルネットの特徴

ニューラルネットは一般に次のような特徴をすべて，或いは部分的に満たします．[4, p.11-12]

(1) 分散性，並列性
互いに同一或いは類似のユニット，つまり**ニューロン**が多数集まり互いに結合して情報をやり取りしながら分散並列的な情報処理を行います．

(2) 局所性
各々のニューロンが得る情報は，他のニューロンから**シナプス結合**を経由して入力される入力信号の状態と，自分自

[5] 本節では特に断らない限り ‘ニューラルネット’ はその人工的な数理モデルを指すものとします．

身の内部の状態，出力信号の状態，また自分の次の結合先のニューロンの状態です．[6]結合のネットワーク自体も局所的であることが多いです．

(3) 荷重の総和，及び活性化関数の非線形性
ニューロンが情報を得るときには，それぞれの入力に対して結合荷重に応じた重み付けを行います．ニューロンの出力は，重み付けされた入力を合計した値そのものであるか，或いはそれを非線形な関数で変換した値です．この関数は**活性化関数**[7]とよばれ多くの場合に飽和的な性質を持ちます．

(4) 可塑性
結合荷重はニューロンが得る情報によって変化します．事前に荷重の値を作り付けておくこともできます．また処理動作をしながらでもその値を更新することもできます．このような動作を一般に**学習**，**自己組織化**といいます．

(5) 汎化性
学習した特定の状況に対して望ましい振舞いをするだけでなく，学習しなかった状況に対しても，学習済みの状況から推測して振舞うことができます．自らの中に連続的な外界の状況地図をつくり，また種々の場合の計量を作ることによって未知状況に対応するころができるのです．**パターン処理**などに応用されています．

参考文献

[1] Symantec, http://www.symantec.com/region/jp/sarcj/data/w/w32.sircam.worm@mm.html, 2001 年

[2] 甘利俊一, 神経回路網の数理, 産業図書, 1978 年

[3] 中野良平, ニューラル情報処理の基礎数理, サイエンス社, 2005 年

[4] 廣瀬明, 複素ニューラルネットワーク, サイエンス社, 2005 年

[5] 脳科学の前線, サイエンス社, 1997 年

[6] 栗田多喜夫, http://www.neurosci.aist.go.jp/ kurita/lecture/neuro.pdf, 2001 年

[6] このような系を一般に相互作用系といいます．これらの素子間に線形則が成り立つことは稀です．
[7] Heaviside 関数やシグモイド関数等が多くのシミュレーションで用いられています．

第II部
JED 活用 Tips

(著者) 0511106 村松雄介

概要
電気通信大学の新入生たちに贈る文章です。電気通信大学情報工学科の計算機室、JED を活用する上で知っておくと便利なことをまとめました。

4 予備知識

4.1 dot file

UNIX 系 OS では、先頭がピリオドで始まるファイルは隠しファイルになります。ls -a で確認することが出来ます。

各ソフトウェアのユーザーに固有な設定は、ホームディレクトリにピリオドで始まるファイル名で記述するのが一般的です。例を挙げると、 csh であれば .cshrc であり、emacs だと .emacs .emacs-faces.el .emacs-fontset.el といったところです。

JED にはサンプルのファイルが置いてあるディレクトリがあります。~sample/ を見てみてください。

4.2 X Window System

X Window System は主に UNIX 系 OS で使われている GUI 環境です。XFree86 や、ライセンスの関係で分離した X.org が主に使われています。ちょっと巨大で分かりにくいという声も聞かれます。

Windows では OS と GUI 環境がくっついてしまっていますが、UNIX の世界では分離されています。

クライアントサーバモデルのつくりになっており、ネットワーク越しに使用できるのが特徴です。

混乱しがちですが、手元のコンピュータで表示を行うのが X server で、遠隔地で実行され X server と通信を行うソフトウェアが X client です。ユーザに近い位置にあるほうが X server です。

5 shell

ほぼすべての OS には、kernel (カーネル) という、文字通り OS の核となるプログラムがあります。OS の基本機能は、この kernel が実行しています。

shell (シェル) の役割は、ユーザーの視点から見た場合に、kernel を包んで見えなくし、すべての操作を shell 経由で行えるようにすることです。

shell とユーザーとの対話はすべて文字によって行われます。ユーザーはコマンドを打つことによって shell に命令を伝え、shell は文字列によってユーザーに応答を返します。

5.1 shell の種類

当然ですが、 shell にはさまざまな種類があります。主に用いられているのは、sh, tcsh, bash の 3 つです。

sh は shell の元祖と言えるもので、Bourne 氏が作ったので Bourne shell と呼ばれます。ほぼすべての UNIX 系 OS で使用することが出来るのが特徴です。シェルスクリプトと呼ばれる簡単なプログラムを書くときに多く使用されます。

tcsh は csh という shell を機能拡張したもので、C 言語に似た文法を持ちます。BSD 系の流れを汲む UNIX で普及しているのが特徴です。

bash は Bourne shell を拡張したものであり比較的後発です。Linux 陣営の人が好んで使用しています。

shell はユーザーの好みが強く反映されるので、特に tcsh と bash において議論が繰り広げられることがあります。

電気通信大学でのコンピューターリテラシーの時間で、tcsh を薦められるか bash を薦められるかで担当教官の好みが判断できるといえます。

表 1: 最低限のキー操作

Ctrl + b	←	カーソルを左へ (backward)
Ctrl + f	→	カーソルを右へ (forward)
Ctrl + n	↓	カーソルを次の行へ (next)
Ctrl + p	↑	カーソルを前の行へ (previous)
Ctrl + a	Home	カーソルを行頭に
Ctrl + e	End	カーソルを行末に (end of line)
Ctrl + d	Delete	カーソル位置の文字を削除 (delete)
Ctrl + h	Backspace	カーソル位置の直前の文字を削除
Ctrl + k		カーソル位置より行末までを削除 (kill)
Tab		補完
Ctrl + d		補完対象の一覧

情報工学科の計算機室である JED では、Linux が入っているにもかかわらず tcsh を薦めてくることから、そういう方が教官を務めていらっしゃると推測できます。

なお、bash も tcsh もできることには大差ありません。なので、初めて shell に触れる方は、 bash と tcsh の両方を触ってみて、好みのものを選ぶと良いでしょう。

もし周りに知識のある人が居る場合は、その人と同じ shell を選ぶことをお勧めします。違う shell だと、質問しても “〜は分からないなぁ” で片付けられてしまいます。また、同じ shell を使っていると親近感が起きるので、やさしくしてもらえるかもしれません。

6 tcsh

ところで、筆者は tcsh 派です。よって、 bash に心を奪われたあなたは、ここからの文章については注意深く読んでください。bash でも使えるテクニックもありますが、大部分は tcsh, csh でのみ使用可能です。

なお、login 時に最初に出てくる shell を変更するには、JED の場合は ldapchsh [8]を使用することで変更可能です。一般的な UNIX 系 OS では chsh コマンドを使用します。

6.1 キー操作

とりあえず、最低限覚えるべきキーを表 1 にまとめました。

これらのキーを覚えると、キーボードのホームポジションから指を離さずに済むため、高速に入力が出来るようになります。Ctrl を多用するため、HHK [9]などの Ctrl が a の横にあるキーボードをお勧めします。[10]

なお、このキー操作の大半は emacs でも使用可能です。出来るだけ早く慣れることを強く推奨します。

6.2 コマンドヒストリ

過去に実行したコマンドをもう一度編集したい場合、Ctrl + p を押すことで過去に実行したコマンドを呼び出せます。似たようなコマンドを実行するときに使うと便利です。

history コマンドで一覧を見ることも可能です。!89 などと打ち込むことで、番号からコマンドを実行することも可能です。

[8] JED の端末は LDAP によって集中管理されているため。
[9] Happy Hacking Keyboard. UNIX 使いが大好きなキーボード。生協でも販売中。
[10] ソフトウェア的に Caps Lock に Ctrl を割り当てる手法も存在します。

表 2: prompt オプション

%%	% そのもの
%n	ログイン名
%~	カレントディレクトリ
%h	ヒストリ番号
%m	ホスト名の最初の. までの部分
%t	AM/PM 12 時間表記時刻
%l	shell の tty
%#	一般ユーザでは”>” スーパーユーザでは” # ”
%B %b	太字開始と終了
%S %s	強調開始と終了

6.3 dot file

tcsh の設定は .tcshrc で変更できます。初めて編集する場合には、デフォルトのものをコピーしてきて編集することをお勧めします。JED の場合、 ~sample/.tcshrc を元に編集することをお勧めします。

.tcshrc は一種のプログラムであり複雑なので、詳細ついては省略しますが、ここではよく使われる prompt と alias および一部のオプションについて解説します。

6.3.1 prompt

プロンプトとは shell が入力を受け付けられることを示す文字列です。tcsh のデフォルトでは “%” が指定されています。ここに有益な情報を表示させてみます。

一時的にプロンプトを変更するコマンドの例は、set prompt = “hoge” です。シェル変数 prompt に hoge という値を設定しています。

shell が解釈する文字列の一部を表 2 に挙げます。すべてのオプションを見るには jman tcsh を参照してください。

気に入ったプロンプトが出来上がったら、.tcshrc に set コマンドを追加しましょう。

シェル変数 rprompt を設定することによって、プロンプト右側にも文字列を表示することが可能です。

参考までに、私の prompt はこんな感じです。

```
set prompt = "%B%n@%m:%l%#%b"
set rprompt = "%~"
```

%B を用いるとプロンプトが目立ちます。複数のコンピュータを使い分けている場合は、%n や %m を用いると区別ができ便利です。%l で tty 番号を表示しておくと screen 使用時に便利です。

6.3.2 alias

alias はコマンドに別名をつけるコマンドです。

alias h history 25 と入力すれば h と打ち込むだけで history 25 が実行されます。

alias の設定は先人の知恵の塊なので、パーミッションが許せば、共同で使っているコンピュータにある先輩の .cshrc, .tcshrc などを覗いてみるとよいでしょう。[11]

[11] 普通は見られるのが嫌な人はパーミッションを変更しています。

表 3: screen のキー操作

C-a C-a	C-a そのものを入力
C-a c	仮想ターミナルの作成
C-a n	次の仮想ターミナルへ
C-a Space	次の仮想ターミナルへ
C-a p	前の仮想ターミナルへ
C-a w	仮想ターミナルの一覧
C-a [0-9]	番号指定で仮想ターミナル移動
C-a S	リージョン分割
C-a Tab	フォーカス切り替え
C-a X	リージョンを閉じる
C-a [	コピー
C-a]	貼り付け
C-a d	デタッチ

7 screen

screen は端末マルチプレクサという、複数のターミナルを切り替えるソフトウェアです。使えば病み付きになります。

複数の kterm を立ち上げて作業しているような状況では、screen を使い複数の仮想ターミナルを切り替えることで劇的に作業効率が改善されます。また、いわゆるコピーアンドペーストを行うことも出来るほか、突然の回線断などからも復旧することが可能になります。

7.1 使い方

screen の使い方は簡単です。ターミナルに screen と打ち込んで Enter を押してください。最初に screen 自体に関する情報が表示されます。すべての仮想ターミナルを exit することで screen は終了します。

表 3 に挙げるキー操作は必須なので覚えてください。ちなみに、C-a は Ctrl + a と同義です。よく使われる表記です。キー操作はカスタマイズすることも可能です。

screen の動作は言葉で説明すると分かりづらいので、コンピュータ上で試しながら読むことをお勧めします。

screen を起動したら C-a c と入力し新しい仮想ターミナルを作成しましょう。C-a n で仮想ターミナルを切り替えてみてください。使い終わった仮想ターミナルは exit コマンドを入力して閉じます。

screen の仮想ターミナル上で screen emacs hoge.txt などと入力することで、新たに仮想ターミナルを作成し、そのなかで emacs を起動することができます。この場合、 emacs を終了すると仮想ターミナルも閉じられます。

C-a S と入力し、上下分割を行えます。S は大文字であることに注意してください。C-a Tab で移動を行い、C-a X で終了です。

C-a [でコピーを行います。カーソルをコピー開始位置にあわせ、Enter を押します。コピー終了位置にカーソルを移動されたら、もう一度 Enter を押します。C-a] で貼り付けを行います。

何らかの作業中に、 C-a d を押すことでデタッチを行います。デタッチを行っても、仮想ターミナル上で動くソフトウェアは影響を受けません。そのまま logout しても大丈夫です。デタッチされたセッションにアタッチし作業を再開するには、screen -r とターミナルに入力します。

外部から ssh などで接続し作業しているときに、回線が切断されるとターミナル上のソフトウェアは終了されてしまいます。しかし、 screen を使って作業している場合は、自動的にデタッチされるので、接続しなおしてアタッチすれば作業を続行できます。

7.2 dot file

デフォルトでは、 C-a がエスケープキャラクターに設定されています。しかし、これは行頭に移動する操作とかぶっているので不便です。

この設定は .screenrc を編集することで変更可能です。エディタで .screenrc ファイルを作成し、内容を “escape ^z^z“ [12]としてください。この例ではエスケープキャラクターが C-z になります。

ここで紹介した内容は screen の一部に過ぎません。jman screen と入力し manpage を参照してみてください。

8 ssh

SSH [13] を使用することで外部から JED のコンピュータを使用することが可能です。

外部から接続可能なホストを以下に挙げます。

- blue99.jed.uec.ac.jp
- red99.jed.uec.ac.jp
- yellow99.jed.uec.ac.jp
- purple99.jed.uec.ac.jp

これらのコンピュータは毎朝 7:00 に再起動されているので、その時間帯は使わないようにしましょう。

UNIX 系 OS を使用しているなら ssh hog99999@blue99.jed.uec.ac.jp と入力するだけで接続可能です。hog99999 は各自の JED のユーザ ID に置き換えてください。そのままで X 転送も可能です。

8.1 PuTTY

Windows [14]のコマンドプロンプトには ssh コマンドは存在しません。

PuTTY [15] というソフトウェアを使用することで ssh 接続が可能になります。日本語化されたもの[16] も存在します。

8.2 X-Deep/32

標準では Windows には X server [17]は含まれていません。そのため、 PuTTY で接続しても X Window System を使うソフトウェアを使用することは出来ません。

多くの Windows 対応 X server は商用でお金がかかりますが、無料で使用できる X-Deep/32 [18] という X server が存在します。X-Deep/32 を起動した上で PuTTY で接続することで Windows でも X 転送により X Window System を使用できます。

PuTTY 設定の、X11 の項目について、“X11 フォワーディングを有効にする” にチェックを入れ、X ディスプレイの場所に “localhost:0” を指定します。

ウィンドウマネージャが無いと使いづらすぎるので、twm & とでも PuTTY 上で実行し、ウィンドウマネージャを起動しましょう。

9 感想

毎度毎度、新入生向けの記事書いちゃってるのがなんとも言えない感じです。締め切りギリギリになって書き始めているので、高度な内容の記事を書く暇が無いです。

一応やっていることとかはあるので、次回こそはしっかりした記事書きたいです。

[12]普通にハットゼットハットゼットです。
[13]Secure SHell
[14]だってゲームが、ゲームがぁぁぁ！
[15]http://www.chiark.greenend.org.uk/ sgtatham/putty/
[16]http://hp.vector.co.jp/authors/VA024651/
[17]表示をするのが X server です。emacs とかは X client です。
[18]http://www.pexus.com/

第 III 部
MMA 新鍵システムの制作

oku

多分電通大で一番手間を掛けて部室のドアを開けているサークル MMA の鍵システムの将来にせまる。

10　鍵システムとは

鍵システムとは、平たく言えばコンピュータで普通のドアの鍵を開閉するためのシステムで、部員全体に物理的な鍵を配布することなく部員全員に任意のタイミングでの部室へのアクセスを可能にする物です。

11　現在の鍵システムのしくみ

鍵システムの物理的な仕組みは比較的単純な物で、モーターを使ってドアのサムターンをまわしているだけです。この部分は先代の鍵システムから継続して使っています[19]

ユーザ認証は FeliCa[20] リーダー (PaSoRi[21]) をドアに設置し、FeliCa の持つ固有 ID[22] に依存し、あらかじめ登録された ID をもつ FeliCa をかざした際に、施錠・解錠動作をトグルで行うようになっています。

以前は部室の外にキーボードを設置して、通常の UNIX ログインを利用していましたが、消防署の指導等の理由で部室の外に物が設置できなくなったため、FeliCa 認証に一本化されています。

12　鍵システムの脆弱性

現在の認証方式は FeliCa のもつオープンな情報[23] を利用しているので、FeliCa 送信機を自作することで突破することが可能です。

つまり「FeliCa そのものを自作することができない」という事実に依存しています。

また、正面突破を避けるなら、外部に公開しているサーバから侵入し、鍵シスを運用しているサーバの権限を奪取すれば良い ‥ 端的に言えば鍵シスのルートパスワードの強度にも依存しています。

もちろん、システムを構成しているマシンの HDD を直接書き換えればなんでも突破できるので部室内部に侵入されれば終了[24] です。

13　新鍵システム

- 鍵の管理が root にしかできない
- 鍵を無効にできるのも root だけ
- ログを外部に出力しない

という現在の鍵システムの問題を解決するために、モダンなデザインの鍵システムを構築中です。

もっとも、FeliCa の IDm を認証に利用するのは変わらないのでエンドユーザとして変わる部分は殆どありませんが。

---

[19] パラレルポートから駆動パルスを直接出してるのは改善したいが。。

[20] FeliCa：Suica や ICOCA、せたまる、Edy のような非接触式 IC カードの形式。電通大では情報工学科の計算機室の認証が FeliCa カードによって行われている。鍵シスではこれらのカードすべてを鍵として登録・利用することができる。

[21] PaSoRi：SONY 純正の PC 向け FeliCa リーダー。公式なサポート OS は Windows のみだが、プロトコルを解析して無理矢理 FreeBSD で利用している。http://libpasori.sourceforge.jp 参照。

[22] IDm：FeliCa のもつ MAC アドレスのようなもの

[23] MAC アドレスや IP アドレスのようなもの

[24] 侵入を検出する手だてを講じた方がいいのかも

13.1 新鍵システムの概要

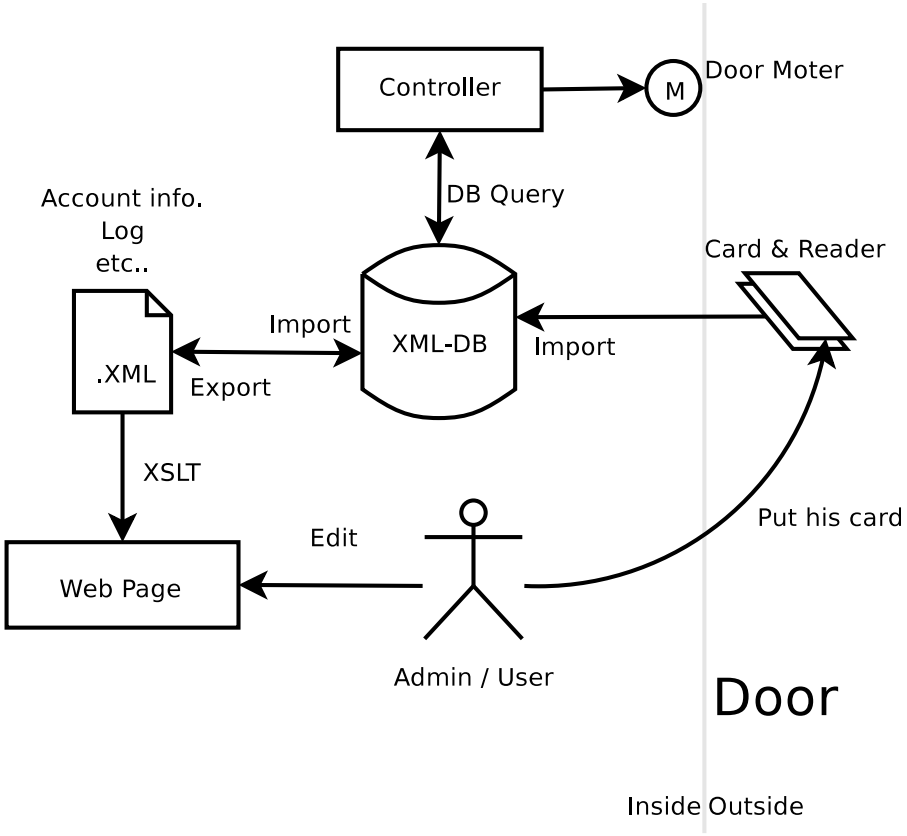

図 1: 鍵システムの概要

13.2 マルチプロトコル XML-DB(niji) の導入

マルチプロトコル XML-DB[25]とは、従来のデータベースの概念を大幅に[26]拡張するデータベースで、ネットワークプロトコルやデバイスドライバを XML で記述することで、データベースが直接デバイスやネットワーク上のホストと対話できるようにするものです[27]。

アカウント情報を既存の passwd やそのほかのファイルに書き並べるのではなく、一つの XML-DB に統合することで、ファイルシステムに依存しない管理を実現するのが目的です[28]。

マルチプロトコル XML-DB の実装として、niji を開発しています。

通常の (ナントカ SQL とか QDBM のような) データベースを用いないのは個人的な趣味が 9 割と、各種プロトコルのためのフロントエンド[29]との接続が面倒なのが 1 割程度で、別に独自の DB を考案するような程の内容ではないです。

Web インターフェースの類は、Ruby on rails とか WebObjects のような MVC モデルベースの Web アプリケーションサーバをいきなり使えば済む話ですが、既存の RDBMS があんまり好きでは無いので却下しました。

[25]XML-DB : XML データベース。XML を格納するためのデータベース。
[26]そして必要以上に
[27]端的に言えば HTTP とか LDAP とか syslog とか話せるデータベース
[28]キーボード認証だけなら NIS なりなんなりで分散すること自体は簡単なのですが。。
[29]HTTP なら管理用 cgi、LDAP なら OpenLDAP のような外部プログラム

13.3 XML によるプロトコル記述 (RAIN) の導入

プロトコル記述のための言語としてたとえば ASN.1 とか XML スキーマのような体系がすでに存在しますが、新たにプロトコル記述様式を必要とするのは主にデバイスドライバの記述のためです[30]。

プロトコル記述をシステムに内蔵せず外部のデータで記述することによって、プロトコル記述を他のアプリケーションで使い回したり、DB のソフトウェアを再構築することなく対応プロトコルを拡張できるメリットが存在します。

14 RAIN

RAIN は、

- 構造体 (C で言うところの struct)
- 文法 (いわゆる BNF 記法)
- 簡単な演算や手続き

を XML で記述するための規約で、この規約にそって記述された XML を DB 側に格納し、DB はそれを参照することでプロトコルの実装を行います。

niji は RAIN による記述を利用して USB 経由の PaSoRi のアクセスや各種ネットワークプロトコル (DNS,LDAP,HTTP) のサービスを提供しています。

現状の RAIN はあまり複雑な動作を記述できないので、鍵がマッチしたらモーターを回すとか、存在しない URI を指定されたら 404 を返すといった実際のプロトコル動作は外部のプログラムで行っています。

15 将来展望

実はこれを書いてる段階でもデモできるレベルに達してないので、まずはデモできる程度の完成度を確保するのが先。。

将来的には、niji にファイルをストレージする機能を持たせて、情報保持を完全に集中させる事を目指したいところです。

また niji は元々PC 上で動作するオペレーティングシステムとして設計・開発されていた物で、niji の思想 (すべてのデータをシステムで統一された構造で管理する事) を元にした OS を作るのが最終目標です[31]。

[30] ASN.1 は BER、XML スキーマは XML のように、データのバイナリ構造に対する依存が存在する。デバイスとの通信を記述するためには任意のバイナリ表現が可能である必要性がある。

[31] 鍵の開け閉めに独自開発の OS が必要なのかは置いとくとして