

WORD

From College of Information Science

メモリ1枚、誰にでもあげる。号

WingoWM

Seiteki Saito Generators Collection

RSAの倒し方

[悲報]ThinkPadを買った結果www

Research Emulator(CORE)入門

SKK紹介チラシ的な何か

もうFLOWERSの話しかしたくない

ココアノロイ



目次

WingoWM	ひだるま	003
Seiteki Saito Generators Collection	あまね	007
RSA の倒し方	れつくす	019
[悲報]ThinkPad を買った結果 www	natto	025
Common Open Research Emulator (CORE) 入門	@rizaudio	029
SKK 紹介チラシ的な何か	ひだるま	042
もう FLOWERS の話しかしたくない	linerlock	043
ココアノノロイ	中原岬	045
WORD 編集部への誘い	鳥	046
編集後記		047

WingoWM

文 編集部 ひだるま

1 はじめに

周りと同じ様な画面を使っているのが気持ち良くない、そんな方はいるでしょうか。僕はそうです。では何を変えるか。Window Manager でしょう。それも使ってる人間がそれなりに限られそうな奴を。

1.1 Window Manager (WM) is 何

多くはデスクトップ環境の一部であり、名前の通り GUI に於けるウインドウの管理を行います。そのままですね。議論を重ねた結果、ArchWiki の記述^{*1}に任せることにしました。

実の所、筆者も良く分かりません。友人に訊いてみても「分からない」との返事だったので分からなくても多分大丈夫です。

1.2 Wingo

Wingo は WM の 1 つです。特徴としては、

- Go 言語で書かれていること
- タイル型 WM として使えること

が挙げられます。タイル型 WM とは「ウインドウが重ならない」「画面全体に表示する」特徴を持つ WM で、xmonad や awesome が有名です。Wingo も、「タイル型 WM として」と書いた通り、ウインドウが重なる、フロート設定にもすることができます。

2 Installation

この記事では Linux のメジャーなディストリビューションを対象としています。筆者の環境は Ubuntu17.04 です。

2.1 Go と Git

先ず、Go 言語と Git をインストールします。やってある場合は飛ばしてください。Ubuntu であれば、

```
$ sudo apt install golang git
```

で入ります。他のディストリビューションでは適宜読み変えてください。

次に GOPATH^{*2}を設定します。好きに設定して構いませんが、ここでは\$HOME/go に設定したものとします。シェルの設定ファイルに以下を追記してください。ちなみにシェルは bash 辺りを想定しています。

^{*1} <https://wiki.archlinuxjp.org/index.php/ウィンドウマネージャ>

^{*2} Go のソースをまとめておく場所のようなもの

```
export GOPATH+="$HOME/go"
```

PATH も通しておきます。

```
export PATH+=":$HOME/go/bin"
```

2.2 Wingo

```
$ go get github.com/BurntSushi/wingo
```

wingo-cmd は設定の変更をするのに必要となります。

```
$ go get github.com/BurntSushi/wingo-cmd
```

2.3 Wingo のショートカットを置く

/usr/bin/以下に wingo という名前で下の様書き、実行権限をつけておきます。

wingo

```
1 #!/bin/bash
2 export GOPATH+=$HOME/go
3 $GOPATH/bin/wingo
```

2.4 デスクトップ環境として設定する

\$GOPATH/src/github.com/BurntSushi/wingo/data/archlinux/wingo-git/にある wingo.desktop を /usr/share/xsessions/ に置き、一度ログアウトすると wingo がログイン画面から選択できるようになります。

入れてあると良いもの

- gmrn: 軽量ランチャー。
- acpi: バッテリー情報簡易表示。
- xterm: 仮想ターミナル。Wingo が想定しているデフォルトターミナルがコレなので。

3 Execution

3.1 実行前に覚えるべきコマンド

後から変更可能ですが、取り敢えず覚えておくと慌てずに済む大事なコマンドです。最初<Mod1>は **Alt**、<Mod4>は Super キー*3 に割り当てられています。

*3 多くの場合 **Windows** キー

Logout

<Mod1>-Shift-C

Terminal(xterm)

<Mod4>-t

3.2 操作を覚える

後述の設定により変更できますが、デフォルトの動作についてもある程度覚えておきましょう。分からなくなったら `$GOPATH/src/github.com/BurntSushi/wingo/config/key.wini` を参照してください。

基本的な操作は以下のようになります。

キー	動作
<Mod4>-Enter	ワークスペースの移動選択
<Mod4>-Shift-Enter	フォーカス中のウィンドウを伴ってワークスペースの移動
<Mod4>-C	フォーカス中のウィンドウの終了
<Mod4>-→	次のワークスペースに移動
<Mod4>-←	前のワークスペースに移動
<Mod1>-F2	gmrn
<Mod1>-a	tile 化
<Mod1>-u	untile 化
<Mod1>-k	次のウィンドウにフォーカス
<Mod1>-j	前のウィンドウにフォーカス
<Mod1>-l	ウィンドウを拡大 (左右)
<Mod1>-h	ウィンドウを縮小 (左右)
<Mod1>-n	ウィンドウを拡大 (上下)
<Mod1>-b	ウィンドウを縮小 (上下)

フォーカスとは、ここではウィンドウを操作対象にすることです。

4 Configuration

先ず、`$HOME/.config/wingo` 以下に設定ファイルを置きます。ターミナルから

```
$ wingo-cmd --write-config
```

を叩くと、`.config/wingo` の場所に `hooks.wini`、`key.wini`、`mouse.wini`、`options.wini`、`theme.wini` が出現します。これらをいじって自分好みの WM にしていくワケです。筆者の場合、<Mod4>-t のターミナルを `urxvt` にしたり、ログイン時にターミナルが表示されるようにするなどしました。`theme.wini` でウィンドウ上部のバー

に於けるフォントの設定ができますが、ここでは TrueType フォントしか指定できないようなので注意しましょう。

筆者の環境では音量調整やバックライトのキーが効かなくなったので以下の様な記述を key.wini に足しました。

key.wini

```
199
200 XF86AudioMute := Shell "amixer -D pulse set Master 1+ toggle"
201 XF86AudioRaiseVolume := Shell "amixer -c 1 sset -M Master 5%+ unmute -q"
202 XF86AudioLowerVolume := Shell "amixer -c 1 sset -M Master 5%- unmute -q"
203 XF86MonBrightnessUp := Shell "xbacklight -inc 10"
204 XF86MonBrightnessDown := Shell "xbacklight -dec 10"
```

5 etc.

Wingo では所謂タスクバー^{*4}が消えるので代替操作を覚えておく必要があります。

nmcli

ネットワークマネージャのコマンドラインでの操作が可能です。

nmtui

ネットワークマネージャのテキストインターフェース版です。nmcli と共に、一部操作（WPA Enterprise の登録など）に未対応なので、別の環境で設定してから Wingo に入ります。

date

日付と時刻が表示されます。

acpi

バッテリー情報が表示されます。acpi パッケージを入れていない場合、upower を調べるなどしてください。

6 おわりに

GPD WIN の画面が剥がれてきたり、落としたショックで microSD カードが懐炉にクラスチェンジしたりした今日この頃。

^{*4} ほら、アレだよアレ、時間とかバッテリー残量とか見えるヤツ

Seiteki Saito Generators Collection

文 編集部 あまね

1 昔話

あなたがインターネットに初めて触れたころ、凝ったページ^{*1}で情報発信をするには FC2 ホームページや忍者ホームページ、あるいは infoseek isweb や Yahoo!ジオシティーズなどのレンタルサーバーサービスを使わなければなりませんでした。

このようなサービスに登録すると、あなたはしばしば FTP サーバーのアカウントを与えられます。あなたはダウンロードしてきた FFFTP をポチポチとクリックし、自分の書いたホームページを `http://example.com/~amane/` や `http://amane.example.com/` といった URL で表示してもらうために、`index.html` と「工事中」と書かれた GIF 画像を（FTPS すら用いずに！）送信していました。

今では無料で独自ドメインを当てられるサービスもあるようですが、あの頃のあなたは `fc2.com` に生やす短くて面白いサブドメインのことばかり考えていましたよね？

そんな牧歌的インターネットを謳歌していたあなたは、ある時以下のような問題に直面しました。

- アングラ感あふれる黒々したテーマを一新し、白を基調とした爽やかホームページにしたいが、各ページに同じ編集を施すのが面倒。
- 全てのページの下に「Copyright (C) Amane Katagiri 2008 All rights reserved.」と入れたいが、同上。
- 編集のたびに `my_homepage_whity_YYYYmdd.zip` という zip アーカイブばかり量産したくない。
- HTML のタグを開いたり閉じたりするのに飽き飽きしてしまった。
- カウンタを置いたり、アクセスを解析したい。
- “自分で掲示板を設置することにした。けれども何をして良いのか分からない。CGI レスキューに救援要請をして本も買った。Perl だ。Perl しかない。しかし Perl がどうして動いているのかは、全く分からなかった。何十行、何百行もの文字の羅列が、どこでどうなって、掲示板になるのか。”^{*2}

多くのことは、スタイルの情報を CSS に分けたり、その場しのぎの小さな JavaScript^{*3}を挿入したりすることでなんとかなりました。しかし、アクセス解析やカウンタのコードの付いたヘッダやフッタをページ間で統一することができなかったり、ページを追加するたびに煩雑な HTML を手書きしなければならない現実、あなたはうんざりしてしまいます。退屈な編集作業に飽き飽きしたあなたのやる気の低迷に合わせて

^{*1} ここではホームページと呼びましょう。

^{*2} `http://anond.hatelabo.jp/20141202220427`

^{*3} たとえば、左上に 2 等身の犬や手裏剣の GIF が表示されるような。

徐々にホームページの更新頻度は減っていき、いくつかのページが「工事中」のままインターネットに黒歴史として漂うことになりました。

同じように生の HTML を編集し、FTP でサーバーにファイルを送りつけていた人々の多くはいつの間にかなくなってしまいました。彼らはどこへ行ったのでしょうか？

その多くは、WYSIWYG^{*4}なエディタで記事を編集できるブログサービスに逃げ出して、もはや HTML のことは忘れました。WYSIWYG が嫌いな人も、Markdown などの軽量マークアップ言語で日々楽しく記事を書いています。今でも hatenablog.jp に生やす短くて面白いサブドメインのことを考えずにはられません。

残りの何割かは、Apache と MySQL、PHP という新しいことばを覚えて WordPress を始めるようになりました。彼らも WYSIWYG なエディタを使って記事をバブリッシュし、プラグインでカウンタを設置してアクセス解析を行うようになりました。ただし、1 人では使いもしない権限管理機能の代わりに、緊急のバージョンアップを適用することと、不定期に記事が改ざんされてしまうことを我慢しています。

Not Found

The requested URL /there/is/no/file.html was not found on this server.

Apache/2.4.10 (Debian) PHP/6.5.0 Server at example.com Port 80

ナイステーマであるところの 404 Not Found

さて、あなたはそのどれにも当てはまらず、静的サイトジェネレーターという選択肢を選ぶことになりました。静的サイトジェネレーターとはなんなのでしょうか？

2 静的サイトジェネレーター

思い出してください。あなたが生の HTML を直接編集して適切なディレクトリに配置するやり方には、いくつか不自由がありました。ページ間の共通要素を正しく配置しなければならないことと、煩雑な HTML タグの記述を強いられることです。それらを解決するために、静的サイトジェネレーターはあなたの代わりに以下のことをやってくれます。

- Markdown や AsciiDoc、reST などの軽量マークアップ言語で書かれた記事を HTML に変換する。
- テーマに沿った共通のヘッダやフッタを各 HTML ファイルに付加する。
- できあがったファイルを適切なディレクトリ構造になるように配置する。

^{*4} What You See Is What You Get という、視覚に頼っている人間の最も邪悪な概念の 1 つ。

一方で、静的サイトジェネレーターは以下のようなことをユーザー任せにします。

- 軽量マークアップ言語で面白い記事を書く。
- 記事をじゃぶじゃぶ書きたくなるような執筆心を煽りまくる WYSIWYG エディタを用意する。

できあがったファイルをサーバーに送信するのは、静的サイトジェネレーターがやってくれるかもしれませんが、あなた自身がサーバーと FTP でおしゃべりしなければならないかもしれません。

ふつうの静的サイトジェネレーターはバージョン管理をしてくれません。バージョン管理が必要であれば、いくつかの設定ファイルとあなたが軽量マークアップ言語で書いた記事を git で管理するとよいでしょう。これまでのように、成果物を日付を記したアーカイブファイルにバックアップしてはいけません。

静的サイトジェネレーターは、突き詰めるとテンプレート的な文字列代入とファイルの配置さえできればよいので、色々なプログラミング言語で作ることができますし、実際に作られています。PHP や m4 は本来そういった処理のための言語の典型でしょう。お気に入りのジェネレーターがあなたが得意な言語で作られていれば、比較的簡単に拡張したり修正したりできると思います。

あなたは好きな言語で書かれた静的サイトジェネレーターを使ってもいいし、知らない言語や嫌いな言語で書かれたものを使ってもいいです。ユーザとしてちょこっと使うだけなら、どの言語で作られているかはあまり関係ありません*5。たいてい、以下の3つが大事です。

1. 使用できる軽量マークアップ言語やその拡張性
2. エレガントなテンプレート文法
3. クールでオリジナリティあふれるテーマ

今回は静的サイトジェネレーターの一覧 (<https://staticsitegenerators.net/>) をもとに、いろいろなジェネレーターを紹介します。

なお、動作確認はすべて ThinkPad X250 (i5-5300U / 16G RAM / 512G SSD) で行いました。ソフトウェアのバージョン情報は以下の通りです。

Debian 4.9.18-1 / bash 4.4.11 / mawk 1.3.3 / sed 4.4

```
$ uname -v
#1 SMP Debian 4.9.18-1 (2017-03-30)

$ bash --version
GNU bash, version 4.4.11(1)-release (x86_64-pc-linux-gnu)

$ awk -W version
mawk 1.3.3 Nov 1996, Copyright (C) Michael D. Brennan
```

*5 処理系をインストールすらしたくないほど嫌いな言語の場合は、動かすのが難しいのでやめましょう。

```
$ sed --version
sed (GNU sed) 4.4
```

テストのために用意した 1200 個の記事タイトルは Moby Project の成果物であるところの Moby Word Lists(<http://www.gutenberg.org/files/3201/files.zip>) から選択しました。

Moby Word Lists

```
$ head -n5 /tmp/NAMES.TXT
Aaberg
Aalst
Aara
Aaren
Aarika
```

3 明日からガンガン使ってほしい静的サイトジェネレーター 6 選

3.1 Pelican (Python 2.7.x or 3.3+)

ドキュメント <http://docs.getpelican.com/>

テーマ一覧 <http://www.pelicanthemes.com/>

プラグイン一覧 <https://github.com/getpelican/pelican-plugins>

テンプレートエンジン Jinja2 (<http://jinja.pocoo.org/docs/>)

Markdown を HTML に変換するやつ Markdown^{*6} (<https://pypi.python.org/pypi/Markdown>)

reST を HTML に変換するやつ docutils (<http://docutils.sourceforge.net/>)

いいところ・とくちょう

- コンフィグが Python スクリプトとして用意されている^{*7}。
- 記事は Markdown か reST^{*8} で書ける。
- 筆者も使っている (<https://ama.ne.jp/>)。

^{*6} 関連拡張パッケージも全部使えて、ドバドバ入れて頑張ると GFM みたいになるのすごい。

^{*7} Python プロダクトにありがち。

^{*8} Python プロダクトにありがち。

インストール / scaffold の作成 / サイトの生成・表示

pip でインストール

```
$ pip install pelican markdown
```

scaffold の作成

```
$ pelican-quickstart
(*snip*)
> Do you want to generate a Fabfile/Makefile to automate generation and
    publishing? (Y/n) y
(*snip*)
```

テスト用記事の作成

```
$ for x in $(head -1200 /tmp/NAMES.TXT); do
    cat << EOS > content/${x}.md
title: ${x}
date: 2017-04-01
# EXAMPLE PAGE
EOS
done
```

サイトの生成・表示

```
$ time make publish
(*snip*)
Done: Processed 1200 articles, 0 drafts, 0 pages and 0 hidden pages in 4.28
    seconds.
$ make serve
cd /tmp/mysite/output && python3 -m pelican.server
```

生成に 4 秒ほどかかりましたね。テストサーバーはポート 8000 番で起動しています。

3.2 Hugo (golang)

ドキュメント <https://gohugo.io/overview/introduction/>

テーマ一覧 <https://themes.gohugo.io/>

プラグイン一覧 <https://gohugo.io/tools/>

テンプレートエンジン [html/template](https://golang.org/pkg/html/template/)^{*9}(<https://golang.org/pkg/html/template/>)

いいところ・とくちょう

- 自分の好きな軽量マークアップ言語で書ける^{*10}。
- WORD 編集部も使っている (<https://www.word-ac.net/>)。

インストール / scaffold の作成 / サイトの生成・表示

go get でインストール

```
$ go get github.com/spf13/hugo
```

scaffold の作成

```
$ hugo new site mysite
$ git clone https://github.com/dim0627/hugo_theme_robust.git themes/
  hugo_theme_robust
```

テスト用記事の作成

```
$ hugo new post/example.md
/tmp/mysite/content/post/example.md created
$ for x in $(head -1200 /tmp/NAMES.TXT); do
    file=$(hugo new post/${x}.md)
    echo "# EXAMPLE PAGE" >> $(echo $file | cut -d" " -f1)
done
```

サイトの生成・表示

```
$ hugo server --theme=hugo_theme_robust --buildDrafts
Started building sites ...
1200 of 1200 drafts rendered
(*snip*)
total in 4852 ms
```

^{*9} golang の標準パッケージ

^{*10} <https://gohugo.io/content/supported-formats/> 外部パッケージと上手くやれるようになっているみたいです。

```
(*snip*)
Web Server is available at http://localhost:1313/ (bind address 127.0.0.1)
Press Ctrl+C to stop
```

5 秒程度かかったみたいです。まあ速い方。

3.3 Jekyll (Ruby)

ドキュメント <http://jekyllrb.com/>

テーマ一覧 <http://jekyllthemes.org/>

プラグイン一覧 <https://jekyllrb.com/docs/plugins/>

テンプレートエンジン Liquid^{*11} (<https://shopify.github.io/liquid/>)

Markdown を HTML に変換するやつ kramdown (<https://kramdown.gettalong.org/>)^{*12}

いいところ・とくちょう

- GitHub Pages が公式に対応している^{*13}ので、乗っかってしまえば全てが上手くいく。
- Markdown が使えます。
- Textile も使えますが、もはや GitHub Pages ではビルドできません^{*14}。そもそも Textile って何？

インストール / scaffold の作成 / サイトの生成・表示

gem でインストール

```
$ gem install jekyll bundler
```

scaffold の作成

```
$ jekyll new mysite
$ cd mysite
$ bundle install --path=vendor/bundle/
(*snip*)
Bundle complete! 4 Gemfile dependencies, 21 gems now installed.
Bundled gems are installed into ./vendor/bundle.
$ sed -i "/^exclude:$/a \ - vendor" _config.yml
```

^{*11} 独自の拡張が入っているみたいです。 <https://jekyllrb-ja.github.io/docs/templates/>

^{*12} 気に入らなければ変更も可能です。

^{*13} 公式に対応しているというのは、ギットハブのサーバーがビルドしてくれるってことで、他人の資源の無駄遣いは楽しい。

^{*14} <https://git.io/v9ADq>

テスト用記事の作成

```
$ for x in $(head -1200 /tmp/NAMES.TXT); do
    cat << EOS > _post/2017-04-01-${x}.md
---
layout: post
title: ${x}
date: 2017-04-01
---
# EXAMPLE PAGE
EOS
done
```

サイトの生成・表示

```
$ bundle exec jekyll serve
(*snip*)
done in 20.033 seconds.
(*snip*)
Server address: http://127.0.0.1:4000/
Server running... press ctrl-c to stop.
```

20 秒かかりました。

3.4 bake (bash)

ドキュメント <https://github.com/taylorchu/baker>

テンプレートエンジン 変数展開と if、each、include、cmd^{*15} が使えるシンプルな実装

Markdown を HTML に変換するやつ Markdown.pl (<https://daringfireball.net/projects/markdown/>)

いいところ・とくちょう

- bash さえあればよい。
- the real static blog generator
- 軽そう。

^{*15} bash スクリプトの実行結果を挿入する。

インストール/サイトの生成・表示

インストール

```
$ git clone https://github.com/taylorchu/baker.git
```

テスト用記事の作成

```
$ ./baker post example
post/2017-04-01-example.md
$ for x in $(head -1200 /tmp/NAMES.TXT); do
    file=$(EDITOR='' ./baker post $x)
    echo "# EXAMPLE PAGE" >> $file
    sed -i "s/draft: true/draft: false/" $file
done
```

サイトの生成・表示

```
$ ./baker bake && cd out/ && python -m http.server
(*snip*)
2017-04-01-example
(*snip*)

real    1m59.597s
user    1m12.960s
sys     0m42.152s
Serving HTTP on 0.0.0.0 port 8000 ...
```

2分かかったので、軽そうっていうのは間違いでしたね。

3.5 mksite (awk, sed)

ドキュメント <https://github.com/clehner/mksite>

テンプレートエンジン 変数展開と navbar の展開が sed で書かれている。

Markdown を HTML に変換するやつ awk で独自実装されている。

いいところ・とくちょう

- depends only on common shell tools

- is fast
- common shell tools であるところの awk の実装の違いによって動かなかったりする^{*16}。
- awk のデバッグは難しい。

インストール / サイトの生成・表示

インストール

```
$ git clone git://celehner.com/mksite
$ cd mksite
$ sed -i 's/md2html := md2html.awk/md2html := ./md2html.awk/' mksite
$ sed -i 's/sub("^" tag/sub("^[*_]"/' md2html.awk
```

md2html にパスが通っていることが前提になっているようなので、レポジトリ内の awk スクリプトを使わせるようにします。

また、一部のバグも修正します。md2html.awk の l.179 の sub に渡った tag に入っている「*」がエスケープされていないのでエラーだと怒られていました。

テスト用記事の作成

```
$ for x in $(head -1200 /tmp/NAMES.TXT); do
    echo "# EXAMPLE PAGE" > ${x}.md
    echo "${x}.html ${x}" >> _nav.txt
done
```

サイトの生成・表示

```
$ time ./mksite
(*snip*)
real    0m10.578s
user    0m0.804s
sys     0m1.108s
```

10 秒でできたので、結構 is fast です。

^{*16} 筆者の環境の awk (mawk) では強調の文法が使えなかったし、busybox のそれではリンクの文法が上手に展開されなかった。

3.6 tclog (Tcl/Tk)

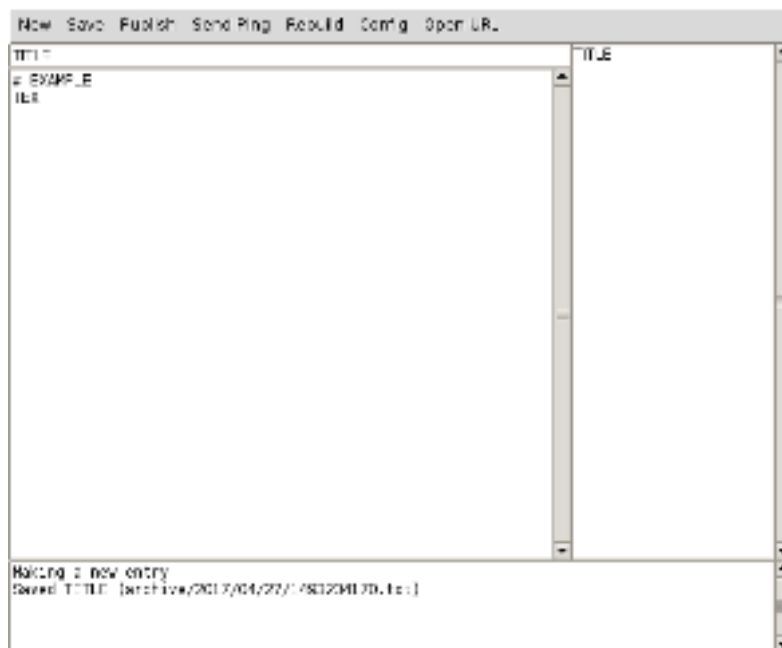
ドキュメント <http://wiki.tcl.tk/9148>

テンプレートエンジン [textutil \(http://wiki.tcl.tk/849\)](http://wiki.tcl.tk/849)

Wiki 風記法を HTML に変換するやつ `ul`、`ol`、`blockquote`、`h3` への変換が `tcl` で独自実装されている。

いいところ・とくちょう

- GUI で楽しくサイトを作れる。
- Win32 バイナリが用意してあったので、Windows でもすぐに使えるかも。
- デプロイと同時に FTP を喋ってくれる。
- あんまり詳細なエラーメッセージを出してくれない。



最高の GUI

インストール / 起動

`tclog` (<https://chiselapp.com/user/dbohdan/repository/tclog/home>) からダウンロードしてください。`tcllib` などを適当に入れる必要があります。

起動

```
$ wish tclog.tcl
```

サイトの生成・表示

1. New で新しい記事を書きます。
2. タイトルを入力してから Save を押すと、記事が `archive` の下に保存されます。
3. Config から接続可能な FTP サーバの設定をします。
4. Publish を押すと Config の情報を用いて FTP でファイルを送信します。

今のところは結果をローカルに生成する機能はないようなので、FTP サーバの設定を間違えると見た目の確認すらできません。

4 まとめ

気になったら Static Site Generators のリスト <https://staticsitegenerators.net/> を見てください。最終的には素直にはてなブログで書いたほうが楽だとも思います。完。

RSA の倒し方

文 編集部 れっくす

1 RSA とは

RSA [1] とは Rivest, Shamir, Adleman の名前から取って名付けられた公開鍵暗号です。名前を聞いたことがあったり簡単な理論は知っている人は多いと思いますが、本記事では RSA はどうすると破ることができるのかという点について解説していきたいと思います。

ちなみに筆者は専門で暗号をやっているわけでもなく数論に精通しているわけでもないので、説明には誤りや不正確な点があるかもしれません。しかも面倒で説明を端折ってる部分も多いので、真面目な解説とは思わず 1 つの読み物として読んでもらったほうが楽しめると思います。

2 基本の話

2.1 mod の世界

(mod N) って書いてあったら常に N で割った余りで計算していると考えましょう。この部分で長い説明をするつもりはないのでいい感じについてきてください。学類の講義だと情報セキュリティがおすすめです。

1 つ注意しておくこととして、加減乗除の計算のうち加減乗についてはいつも通りの感覚でいいのですが、除算だけは mod 上で扱いが異なるので注意してください。どういうことかということ、mod というのは余りのことなので整数での計算になるのですが、それでは $3 \div 5$ みたいな計算ができなくなってしまいます。割り算をするときは逆元という概念を導入して 3×5^{-1} といった具合に計算する必要があります。この方法も説明すると長くなるのでうまいことやってください。

2.2 オイラーの定理とオイラーの ϕ 関数

RSA の話をするときは普通「大きい素数を持つ数の素因数分解は大変ですね～」とか「公開鍵・秘密鍵はこういう風に値を決めましょう」みたいな話から始まりますが、本記事ではもっとコアな部分からスタートします。

読者の中には高校生の頃にフェルマーの小定理という定理を習った人もいると思いますが、ここで紹介するオイラーの定理はそれを少し拡張したのになります。フェルマーの小定理を思い出してみると、 p を素数、 a を p と互いに素な整数としたとき、

$$a^{p-1} \equiv 1 \pmod{p} \quad (1)$$

が成立するという話でした。次にオイラーの定理を見てみましょう。素数 p を 2 以上の整数 N に置き換え、

a を N と互いに素な整数とした時に、

$$a^{\phi(N)} \equiv 1 \pmod{N} \quad (2)$$

が成り立つというのがオイラーの定理になります。ここで $\phi(N)$ すなわちオイラーの ϕ 関数が出てくるわけですね。記号的にはファイですがトーシェントと呼ばれるそうです。なぜそう呼ぶのかは詳しく知らないのですが数学科のお友達に聞いてみてね。 ϕ 関数が何をするのかというと、1 から与えられた整数 n までの間で、 n と互いに素な整数の個数を返します。 $\phi(p)$ のときは p 個の数の中で p と互いに素でない数は p だけなので $\phi(p) = p - 1$ になります。フェルマーの小定理と同値になっていてなるほど拡張というわけですね。

また、互いに素な a, b があつたとき、 $\phi(ab) = \phi(a)\phi(b)$ になります。これ意外と重要です。

2.3 鍵を作る

いよいよ 2 つの素数をぶつけて鍵を作ります。

2 つの異なる素数 p, q を用意してその積をモジュラス N とします。

$$N = pq \quad (3)$$

RSA として有名なこの式はそんなに重要じゃなくて、次の e, d の決め方が実にうまいんですね。どういう風に決めるかというとうこうです。

$$ed \equiv 1 \pmod{\phi(N)} \quad (4)$$

ここで、 e というのは公開鍵のパラメータで d は秘密鍵のパラメータです。 e の代表的な値としては 3, 257, 65537 があります。繰り返し 2 乗法で計算を高速化でき、かつ素数なので $\phi(N)$ と互いに素になりやすい数という話です。 d は e と掛けて 1 になる値にしなければならないので、 e の逆元ということになります。具体的な値を想像できたところで式 4 を見てみると、余りを使う計算なので次のような表現ができます。

$$ed + k\phi(N) = 1 \quad (k : \text{整数}) \quad (5)$$

さっき紹介したオイラーの定理 (式 2) とこの式 5 でうまくやるといい感じになります。

$$\begin{aligned} a^1 &\equiv a^{ed+k\phi(N)} \pmod{N} \\ &\equiv a^{ed} a^{k\phi(N)} \pmod{N} \\ &\equiv a^{ed} (a^{\phi(N)})^k \pmod{N} \\ &\equiv a^{ed} \pmod{N} \end{aligned}$$

というわけで $a = a^{ed} \bmod N$ ってことができて便利なんですね。

あ、そうそう。忘れてましたが公開鍵は (N, e) で秘密鍵は d です。

2.4 暗号化

暗号化したい文（平文）を数値化したものを M 、暗号文を C として、

$$C = M^e \bmod N$$

こう。

2.5 復号

みんな知ってるよね。

$$\begin{aligned} C^d \bmod N &= (M^e)^d \bmod N \\ &= M^{ed} \bmod N \\ &= M \end{aligned}$$

3 RSA を倒す

ここまでのお話はこれのための茶番でした。前提知識として必要なことだけど書くのが面倒だし、真面目に書くと超長くなるので書いて嫌になった。ということなのでここからが本番です。RSA は“普通は”安全なんですが、一部状況下や使い方を誤ったときに強度が弱くなってしまうことがあります。以下ではそのような状況下で RSA を倒す方法を紹介します。

3.1 $M^e < N$ のとき

これは M^e の値が N より小さくてそのまま e 乗根を取れば復号できてしまうケースです。 $e = 3$ とか e が小さいときに起こりがちです。

$$\begin{aligned} (N, e, M) &= (983288011, 3, 38) \\ C &= M^e \bmod N = 54872 \\ C^{\frac{1}{3}} &= 38 \end{aligned}$$

3.2 N_1, N_2 で同じ素因数を含む場合

3つの素数 p, q, r を使って、

$$\begin{aligned} N_1 &= pq \\ N_2 &= pr \end{aligned}$$

のように表すことのできる場合、 N_1 と N_2 の最大公約数を取ると p が出てきて便利です。最大公約数の計算はユークリッドの互除法を使えば一瞬ですね。

3.3 同じ平文を異なる N で暗号化しているとき

ある平文 M に対して、 N_1, N_2, \dots, N_i と複数の異なるモジュラスで暗号化している暗号文を入手できた場合、中国剰余定理を用いてモジュラスを拡張することができます。中国剰余定理というのは大昔の中国の算術書に『ある数を 3 で割ると 2 余る。5 で割ると 3 余る。7 で割ると 2 余る。ある数はいくつか。』という問いとその求め方が書かれていたことから来ています。

暗号文 C_1, C_2, \dots, C_i として、次のような i 個の式が成り立つとします。

$$C_1 = M^e \pmod{N_1}$$

$$C_2 = M^e \pmod{N_2}$$

$$\vdots$$

$$C_i = M^e \pmod{N_i}$$

このとき、中国剰余定理を用いると、

$$C = M^e \pmod{N_1 N_2 \cdots N_i}$$

を満たす C を計算することができます。モジュラスが大きくなったことにより $M^e < N_1 N_2 \cdots N_i$ となる場合には e 乗根を取れば M を取り出すことができます。 M^e のビット数は M のビット数の e 倍程度になるため、 $M < N_i$ から大体 e 個の暗号文を集めると復号するのには十分です。

3.4 $M_2 = \alpha M_1 + \beta$ のとき

これは 1996 年に Coppersmith ら [2] によって発表された攻撃方法の一部で、 M_1, M_2 はわからないけど M_1 と M_2 の間の関係が分かる場合に使える方法です。上の式だと α, β がわかっている状況ですね。わかりやすさのために $M_2 = \alpha M_1 + \beta$ と書きましたが、一部例外があるものの係数がわかっている多項式関数 P を用いて $M_2 = P(M_1)$ で表現できれば大丈夫っぽいです。

この時に何が起きてるかというと、単純に 1 つの未知数に対して 2 つ式ができることになるんですね。嬉しくありませんか？

具体的には、

$$x^e \equiv C_1 \pmod{N}$$

$$(\alpha x + \beta)^e \equiv C_2 \pmod{N}$$

となる x を探すわけなんですけど、これを少し書き換えて……

$$x^e - C_1 \equiv 0 \pmod{N} \tag{6}$$

$$(\alpha x + \beta)^e - C_2 \equiv 0 \pmod{N} \tag{7}$$

こう。そして高校の頃の数学を思い出しましょう。

$$a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n = 0$$

という方程式があった時、もし $x = x_0$ を解に持つならば、左辺を因数分解すると $x - x_0$ の項が現れますよね。まさにそういうことが今起きていて、式 6,7 の左辺では因数分解した時に $x - M_1$ の項が現れる状態なわけです。あとはどうにかして $x - M_1$ を取り出せば終わりです。これは多項式の公約数なのでここでもユークリッドの互除法が活躍します。

そう、ユークリッドの互除法です。ユークリッドの互除法は最大公約数を求めるアルゴリズムですが、多くの場合最大公約数が $x - M_1$ になるということなので式 6, 7 の左辺同士でユークリッドの互除法を適用すると求めている式が手に入ります。ちなみに割り算は通常の割り算とは異なるので人間が手計算するのには向いていません。SageMath や Mathematica などの強い数式処理ソフトの力を借りましょう。

最後に今まで説明した内容を自動的にやってくれる SageMath のコードを置いておきます。詳しい人は使ってみてください。ちなみにこのコードだと e が大きくなったときに計算量が増えるワカメ^{*1}するので気をつけましょう。

Listing 1: 自動で頑張ってくれるやつ.sage

```

1 p = random_prime(2^256)
2 q = random_prime(2^256)
3 N = p * q
4 e = 257
5
6 a = 3
7 b = 1234
8 m = 123456789012345
9
10 PR.<x> = PolynomialRing(Zmod(N))
11
12 g1 = x^e - m^e
13 g2 = (a*x+b)^e - (a*m+b)^e
14
15 def mygcd(f1, f2):
16     while f2:
```

^{*1} $O(e \log^2 e)$ のやつもあるらしいけど頭が悪いので詳細は不明です

```
17         f1, f2 = f2, (f1 % f2)
18     return f1.monic()
19
20 g = mygcd(g1, g2)
21
22 print('Message: {}'.format(m))
23 print('Decrypted: {}'.format(-g[0]))
```

4 まとめ

本記事では RSA の簡単な紹介と弱い RSA の倒し方について紹介しました。筆者はこういうテクニックを駆使して CTF で出題される暗号問題をちまちま解いています。ただ、最近難化が激しくこの程度の知識じゃほとんど太刀打ちできないんですよね。

ここで紹介した以外にも様々な倒し方がありますが、大部分の攻撃方法については Dan Boneh [3] によりまとめられているので読んでみるといいかもしれません。

参考文献

- [1] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” vol. 21, pp. 120–126, feb 1978. Commun. ACM.
- [2] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, *Low-Exponent RSA with Related Messages*, pp. 1–9. Advances in Cryptology, EUROCRYPT’ 96, Springer-Verlag, 1996.
- [3] D. Boneh, “Twenty years of attacks on the rsa cryptosystem,” vol. 46, no. 2, pp. 203–213, 1999. Notices of the American Mathematical Society (AMS).

[悲報]ThinkPad を買った結果 www

文 編集部 natto

ブンブーンハローゆーちゅーぶ！ どうも natto です。

情報科学類生はパソコンが命！ 僕もねついに買っちゃましたよ。

それがこちら

ThinkPad T460s

イエー——イ、フーフ——、パチパチ

僕は今まで MacBook を使ってきたんですけど YouTube をはじめるなら ThinkPad と皆が言っていたので買いました。パソコンのことよく分らないんですけどメモリは 20GB 搭載しています！ それにしてもかっこいいですねこのオタクブラックとワンポイントで使われている赤、一般的なシルバーの筐体よりかっこよくてドヤれますね。

じゃあここでは前使っていたボンコツ MacBook と新品の ThinkPad をぶつけてみましょう。ThinkPad の売りである堅牢性はいかほどなのかこの現実のみを放送する YouTube で実験するわけですね。

1 正直な話

書いている自分が一番つまんないと思ってる。

2 BunBun Hello Awesome

WingoWM に加えて僕もウィンドウマネージャ (WM) の話をします。

3 awesome is awesome

awesome^{*1}とは最近大流行りのタイル型ウィンドウマネージャです。原宿のねーちゃんもこれの話しかしてないよ。

以下信用できるソースから引用

"awesome は高度に設定可能な、次世代の X フレームワークウィンドウマネージャです。高速に動作し、拡張しやすく、GNU GPLv2 ライセンスを使っています。awesome はパワーユーザーや開発者など、日常的に PC を使い、きめ細かいグラフィカル環境を求めている全ての人をターゲットにしています^{*2}。"

^{*1} <https://awesomewm.org>

^{*2} <https://wiki.archlinuxjp.org/index.php/Awesome>

awesome だと思っている点は

- 設定ファイルは lua だが lua を知らなくても高度な設定が可能である点
- スナック菓子のように軽い
- WM 自体がマウスを必要としていない
- タグ（仮想デスクトップのようなもの）があり論理的に切り替えができる

4 導入

linuxOS をお使いのあなた幸運なことにすぐに awesome できます。pacman、apt、rpm とかでインストールできます、そうしたらログインマネージャとか xinitrc に `exec awesome` で起動できますね。Windows や macOS を使用している人は残念ながら使用できません^{*3}。これでは awesome な気分にならないのでこの際 linux デビューしちゃいましょう。

5 基本操作

ウィンドウマネージャとしての機能はすべて modkey（一般的には windows キー）で操作できます。これにより WM 自体はポインタデバイスを必要としません。

下表に基本的な操作を示します。awesome ではウィンドウとは Xclient のこと、タグは上記の通り仮想デスクトップのようなものです。

^{*3} macOS は awesome をインストールできますが X11 上で動作するアプリケーションしか操作できません。

modkey + j	フォーカスを次のウィンドウに移動する
modkey + k	フォーカスを次のウィンドウに移動する
modkey + Shift + c	ウィンドウを閉じる
modkey + f	フルスクリーン化
modkey + m	ウィンドウを最大化
modkey + Control + m	ウィンドウの最大化を解除
modkey + n	ウィンドウを最小化
modkey + Control + n	ウィンドウの最小化の解除
modkey + space	ウィンドウレイアウトを切り替える
modkey + Shift + j	ウィンドウの配置を前のウィンドウと入れ替える
modkey + Shift + k	ウィンドウの配置を次のウィンドウと入れ替える
modkey + l	ウィンドウを右方向にリサイズする
modkey + h	ウィンドウを左方向にリサイズする
modkey + [1-9]	指定のタグに移動する
modkey + Shift + [1-9]	指定のタグにウィンドウを移動する
modkey + Return	ターミナルを起動する
modkey + r	ランチャを起動する (アプリケーションの起動)
modkey + s	ヘルプの表示
modkey + Control + r	awesome を再起動する
modkey + Shift + q	awesome を終了する示

だらだら書いたけど自分で使ってみるのが一番です。また上記の設定は自由に変更、追加できます。

6 9回裏2アウト

うーんそろそろ時間がなくなってきたのでこの話はまた次回にしましょう。まだはじまったばかりだって心配御無用です。”次号でがつつり書きます。

7 話を戻すと

という訳でパソコンを買ったのでパソコンがひとつ余ってしまいました。

そ・こ・で！！ ゆーちゅーばーになりたい人は僕に申し付け下さい。僕の使っていた MacBook を動画編集用にプレゼントします。

8 さいごに

もってるパソコンを見せびらかすのではなく中身で勝負するべきだと思います。
hikakin だってそうしてる。

9 P.S.

実際 Macbook と thinkpad をぶつけ合うと Macbook が凹みました。

Common Open Research Emulator(CORE) 入門

文 編集部 @rizaudou

1 はじめに

最近知り合いにネットワークエミュレータの話をしたら、大変意外なのですが全く知らないということで驚きました。そこで今回は easy-to-use GUI や efficient and scalable という要素を持つネットワークエミュレータ、CORE を紹介したいと思います。恐らく GNS3^{*1} や ns-3^{*2} などは有名なので皆さんご存知かと思いますが、CORE を知っている方はまずいないのではないのでしょうか。

1.1 対象読者

この記事ではネットワークエミュレータとは何か分かっていて単純な LAN 構築の知識があり、ネットワークを構成する機器の大まかな役割を理解している人を対象とします。

読むことをおすすめしたい人

- ネットワークの知識をつけたいが、座学だけでは物足りないという人。
- ネットワーク構築のプロトタイピングツールを探している人。
- 面白い物が好きな人。
- potetisensei^{*3}。

1.2 注意書き

前提知識がかなり必要な上、CORE は解説しなければどこも嵌まる部分が多いのでワイアレスネットワークに関しては今回の記事からは省きます。主要な MANET とかやりたい方は WORD 編集部に来て私を探るか、ドキュメントを読んでください。ドキュメント読む方が簡単かつ重要だと思います。

また、これでもドキュメントに書いてあることは大体意識したのですが、仕組みを知りたいという方はドキュメントそのものや CORE に関連する論文を読んでください。参考文献の所に主なソースは記載しておきました。

セキュリティ上の注意

このツールは内部で実際のパケットを生成しているため、不用意に物理インターフェースを内部で使うと外部に大量のパケットが伝送される可能性があります。使用時は管理下のネットワークで十分な安全策を取った上で行ってください。問題がなければ仮想マシン上で NIC をホストコンピュータとだけの接続に限定することや、当たり前ですが必要がなければ IANA が確保しているプライベートアドレスを使うと良いでしょう。

^{*1}<https://www.gns3.com/>



^{*2}<https://www.nsnam.org/>

^{*3}読んで

2 Common Open Research Emulator(CORE) #とは

Common Open Research Emulator^{*4} とは、NRL 内 NCS ブランチでボーイング社の人たちが作成したネットワークをエミュレートするツールです。凄い挑戦的な名前ですね^{*5}。BSD-2 ライセンス下で提供されているオープンソースソフトウェアで、現在の開発は <https://github.com/coreemu/core> で行われているようです。GUI もしくはシナリオファイル、Python スクリプトでエミュレーターの制御ができます。余談ですが、CORE は NRL で開発された物が依存関係に多く入ってます。自組織大好きなんですかね？

2.1 話はどうでも良いから触りたい人向け QuickStart

VM 入れて X11 Forwarding over SSH^{*6} して、core-gui を叩く。GUI が出たら   で *sample* なんか *imm* を開いて遊べる。後は多分ノリと気概で行けるんじゃないかな。頑張って穴に嵌ってください。

2.2 仕組み

CORE のノードはライトウェイトな仮想マシンで出来ています。Linux と FreeBSD ではそれぞれ別の仮想化方法で動きます。

- Linux では network namespaces(netns や LXC といった方がわかりやすいと思いますが) でノードを作成し、Linux Ethernet Bridging で仮想ネットワークを構成しています。
- FreeBSD では Jails と netgraph でやってます。

ノードやリンクのエミュレートはこれで動かすのがデフォルトですが、ENAME^{*7} や ns-3 をエミュレートに使う事ができたりします。

2.3 導入

CORE では VM イメージ^{*8}を提供しているので、本記事ではこれを推奨します。気に入ってもっと深く使いたいという人は CORE ドキュメントの Install についての部分を読んでください^{*9}。

VCORE インストール

公式のファイルサーバー (<https://downloads.pf.itd.nrl.navy.mil/core/>) で VCORE^{*10} という仮想マシンイメージが配布されています。vmware-image^{*11} フォルダ内の *vcore-4.7.zip* をダウンロードして個々の好きな VMM、VirtualBox や VMware Server とか QEMU で動かしてください。

^{*4}<http://www.nrl.navy.mil/itd/ncs/products/core>

^{*5}「われわれはかしこいので」とか言ってそう

^{*6}ssh -Y でやる

^{*7}<http://www.nrl.navy.mil/itd/ncs/products/emane>

^{*8}4.7 と古いですが

^{*9}<http://downloads.pf.itd.nrl.navy.mil/docs/core/core-html/install.html#vcore>

^{*10}名前付けがださい！

^{*11}ファイルの中には vmware 以外のイメージもあるぞ！！！！

パッケージからのインストール

記事では VM を使うのですが、Tips としてパッケージからのインストール方法も書いておこうかと思います。Fedora や CentOS などでもパッケージが存在するらしいのですが、オフィシャルなりボジトリからは見つからなかったので今回は Ubuntu か Debian でのパッケージインストールだけ紹介します。

```
apt-get install core-network quagga isc-dhcp-server
```

でおそらく全て入るはずです。

ちなみに無線ネットワークに関してはこの手順だけでは高品質なエミュレーションにはなりません。詳しくはマニュアルを読んでください。そこまでやる人は大体研究者か好き者なので言わなくてもマニュアル読んでそうですが...

仮想マシンへの接続方法

2 つ方法があり、仮想マシンを直接使う方法と X11 Forwarding over SSH があります。macOS を使っていて X11 Forward が面倒などの理由が無ければ、ssh -Y [hostname] などとして使いましょう*12。

導入・テストでの FAQ

Q. サービスが動かない

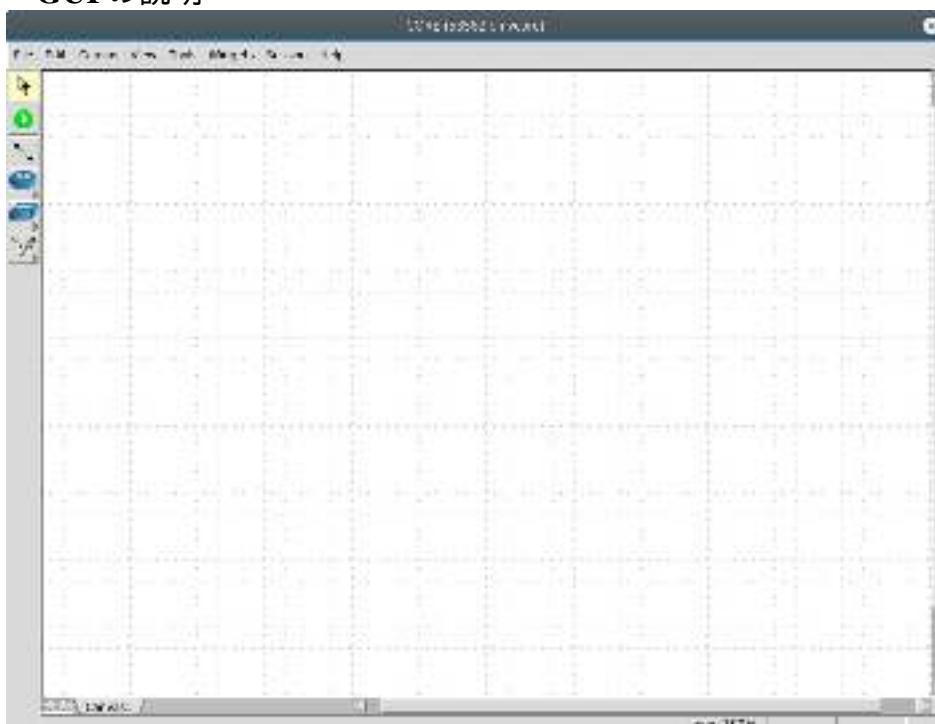
A. サービスが CORE ホストにインストールされてないと動きません。デフォルトでは大体入ってないっぽいので入れましょう

Q. エミュレーションが正常に行われてないが、理由がわからない

A. 後述する Check Emulation Light を使ってエラーを発見してください

*12 ここで-Y、つまり Trusted なものにするのは Untrusted だと core-gui が動かないからです。もしもこの仮想マシンが信用できない場合は直接使ってください。

3 GUIの説明



上の画像が core-gui の画面です。なんとも Tcl/Tk っぽいですが、我慢してください。この GUI はわかりにくいところがあるのでこの節では念入りにマニュアル [1] に沿って説明を書いていきます。GUI にはプライマリモードとして **Edit** モードと **Execute** モードがあります。通常オプション無しで core-gui を動かした場合、Edit モードが初期モードになります。一応他にも Batch モードがあったりするんですが、操作が難しいので割愛します。

3.1 Editing ToolBar

Selection Tool ノードを選ぶ時、移動する時、設定する時に使ったりします。

Start button Execute モードに移行します。

Link 2つのノードをクリック&ドラッグでリンクできます。

Network-layer virtual nodes

MDR, PRouter は今回は説明が複雑になるのでしません。

Router Quagga OSPFv2 と OSPFv3 が動いてパケットのルーティングをしてくれます

Host サーバー役、デフォルトルートを持っていて SSH サーバーが動いています

PC PC です。デフォルトではプロセスは何も動いてません。

Edit CORE Node Types ダイアログを呼び出し、新しいノードタイプを作成できたり、新しいノードのここでそれぞれのノードタイプのデフォルトサービスを変更できます。

Link-layer nodes

Hub リピータハブです。来たパケットは他の繋がっている全てのノードにフォワードします

Switch スイッチです。持っている Ethernet アドレスのハッシュテーブルを使って 来たパケットが来たホストにフォワードします

RJ45 CORE が動いているホストの物理インターフェースを表すノードです。設定されたインターフェースは Execute モードになった際にアドレスが削除されコントロールが CORE に奪われます。そしてプロミスキャスモードに変更されます。

Tunnel Tunnel ノードは GRE トンネルを使って他の CORE エミュレーションと接続することができます。

Annotation Tools

Marker キャンバスにマークを描くことができます

Oval キャンバスに円を描くことができます

Rectangle キャンバスに長方形を描くことができます

Text キャンバスにキャプションを描くことができます

3.2 Execution ToolBar

Selection Tool Execute モードでは Selection ツールはノードをキャンバス内で動かすのに使います。ノードをダブルクリックでそのノードのシェルウィンドウが開きます。

Stop button エミュレーションを終了し、Edit モードに移行します

Observer Widgets Tool 有効になっている際にノードにマウスオーバーすると、そのノードに対しての情報がポップアップされます。

Plot Tool リンクをクリックするとスループットウィジェットが起動し、リアルタイムでトラフィックスループットのプロットが表示されます

Marker フリーハンドでキャンバスに線を描くことができます、線は保存されません。

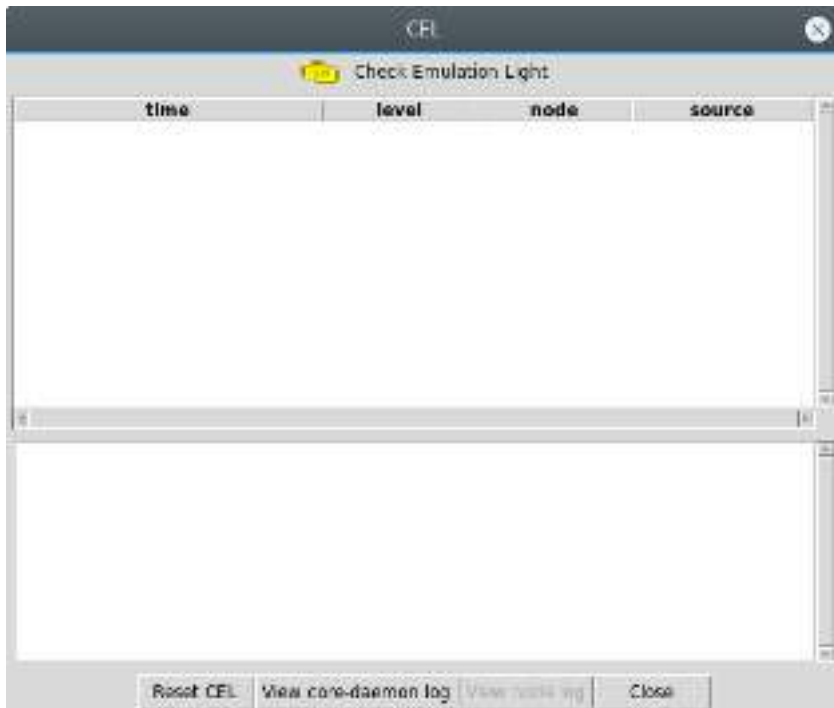
Two-node Tool 始点と終点のノードを選び、一回のみ *traceroute* もしくは */ping -R/*を実行できます。

Run Tool このツールはシェルコマンドを全て、もしくは一部のノードで動かすことができます。リストボックスでノードを選択でき、テキストボックス*¹³でコマンドを入力できます。

*¹³ この言い方は怪しいですが

3.3 Check Emulation Light

Check Emulation Light(以下、CEL) は GUI の右下隅にあります。これはエミュレーション中、問題が発生した際に黄色のアイコンが表示され、クリックすると CEL ダイアログを表示します。



CEL ダイアログは CORE daemon から受け取った例外のリストを格納しています。例外は時間, レベル, ノードナンバー, ソースを持っています。CEL が点滅をしている時は一つ以上の fatal レベルの例外があるという事を示しています。fatal レベルの例外が発生した場合、一つ以上のエミュレーションの基本要素が作成されなかったという事になります。

4 エミュレーション

それではエミュレーションについての話に入っていきます。シナリオファイルは.imn もしくは.xml という拡張子で保存されます。GUI の設定やシナリオファイルは `~/core/` に保存されます、新しいノードタイプを追加する際などはこのフォルダをバックアップするようにしておくといいでしょう。

さらに、Python によって制御ができるようになっているため、例えばマニュアルにある以下のようなスクリプトで厳密に制御することが出来ます。

```
#!/usr/bin/python

from core import pycore

session = pycore.Session(persistent=True)
```

```
node1 = session.addobj(cls=pycore.nodes.CoreNode, name="n1")
node2 = session.addobj(cls=pycore.nodes.CoreNode, name="n2")
hub1 = session.addobj(cls=pycore.nodes.HubNode, name="hub1")
node1.newnetif(hub1, ["10.0.0.1/24"])
node2.newnetif(hub1, ["10.0.0.2/24"])

node1.icmd(["ping", "-c", "5", "10.0.0.2"])
session.shutdown()
```

4.1 エミュレーションの前に

エミュレーション内のサービスは外部ソフトウェアを使っているため、インストールが必要です。VCOREを使っている人も含め、以下のコマンドで今回使うサービスはインストールしておいてください。

```
sudo apt-get install quagga
sudo apt-get install openvpn
sudo apt-get install isc-dhcp-server
```



注意として目に見える問題が起きても起きなくてもエミュレーションを開始した際には CEL が何か例外を受け取っていないか確認をしたほうが良いでしょう。

4.2 サンプルシナリオを動かす

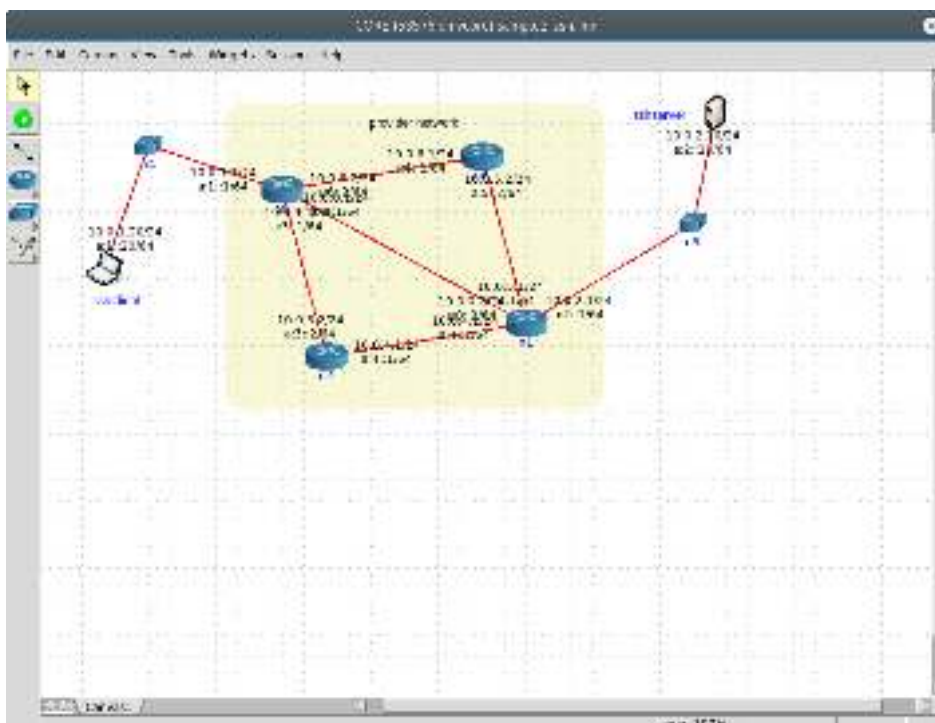
ここからは CORE でサンプルシナリオを使って、CORE の雰囲気を感じてもらいたいと思います。まずデーモンを以下のコマンドで動かします。

```
sudo /etc/init.d/core-daemon start
```

その後、*core-gui* を動かして GUI の画面を表示させます。

では VCORE に搭載されているサンプルシナリオを試していきましょう。  から `~.core/config/sample2-ssh.imn/`^{*14}を開きましょう。開いたら以下の画像のような物が出てくるでしょう。

^{*14} VCORE にしかないサンプルシナリオです



出てきましたね？ では ■ を押してエミュレーションを開始しましょう。開始するとそれぞれのノードに四角い枠が出た後、色が変わり消えたと思います。これはそのノードの作成が終わったという事です。

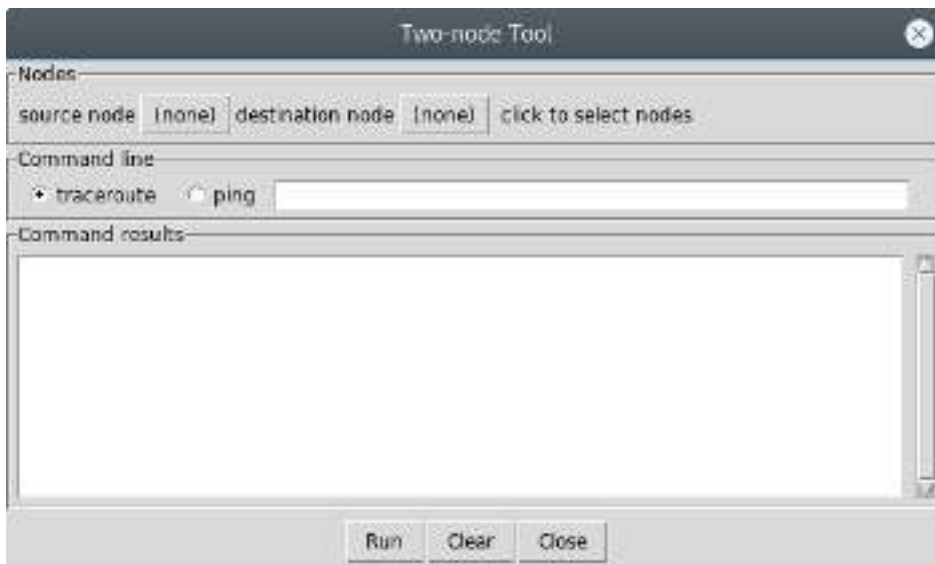
では `sshclient` という名前のノードをダブルクリックしてください。そうするとそのノードのターミナルが出てきます。

これは皆さんが普段使うターミナル・シェルと同じ物^{*15}です。試しに `ping` や `traceroute` を打ってみましょうか。

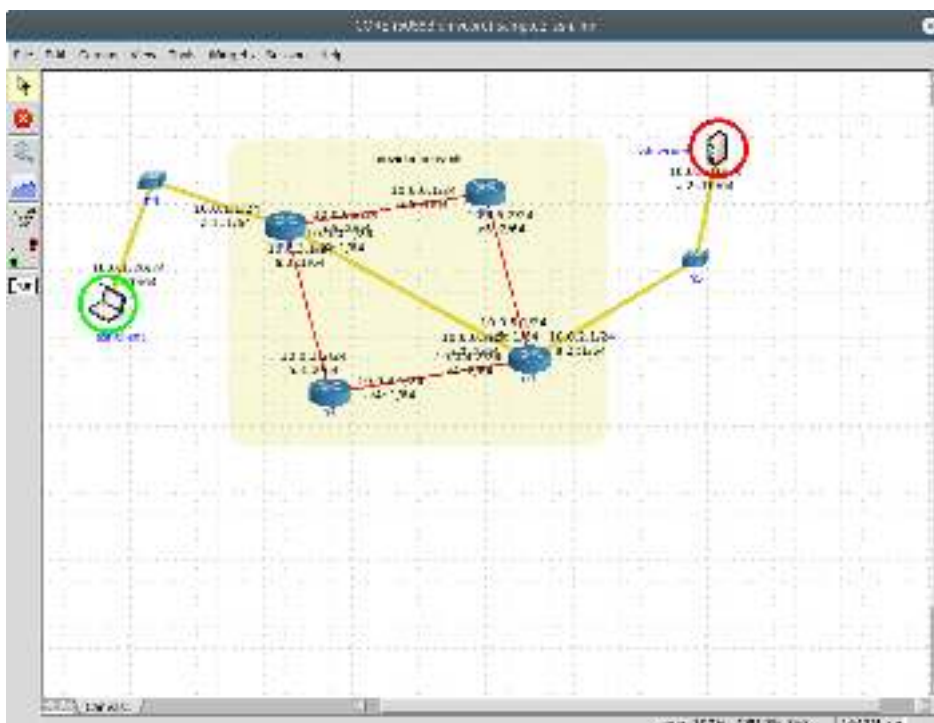
```
root@sshclient:/tmp/pycore.40674/sshclient.conf# traceroute 10.0.4.1
traceroute to 10.0.4.1 (10.0.4.1), 30 hops max, 60 byte packets
 1  10.0.1.1 (10.0.1.1)  0.107 ms  0.018 ms  0.013 ms
 2  10.0.0.2 (10.0.0.2)  0.047 ms  0.021 ms  0.020 ms
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * 10.0.4.1 (10.0.4.1)  0.041 ms  0.010 ms
```

^{*15} 完全に同じかと言われるとちょっと違いますが


確認したいところまで通ってますね？ これはシェルからやりましたが、CORE の機能でこの二つは可視化された上で提供されています。試してみましょう。■ から twonode ツールを起動しましょう。

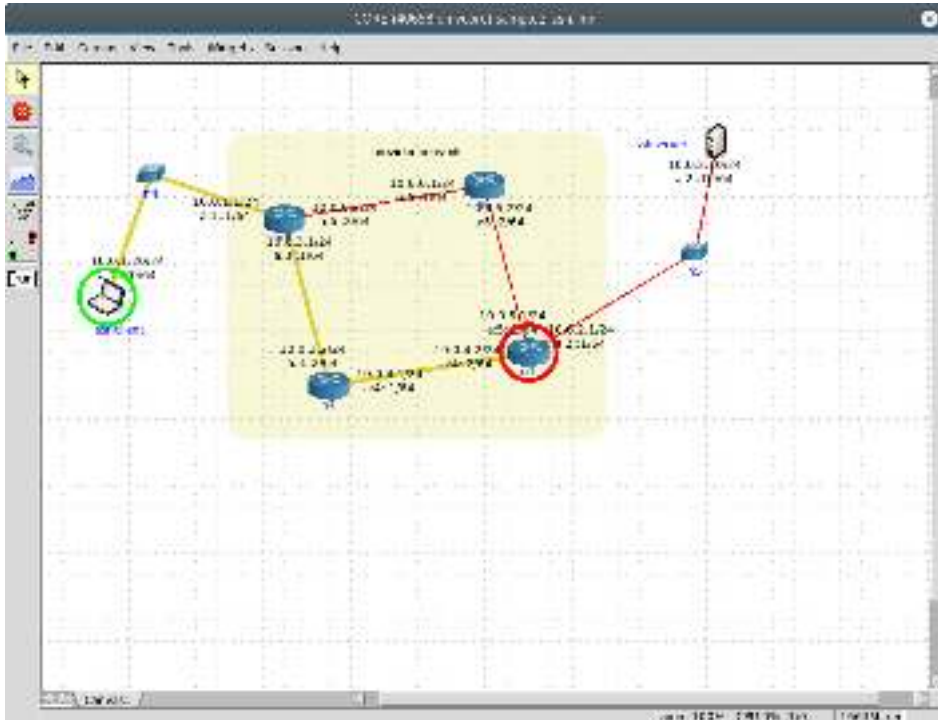


このツールでは traceroute は明示的に経路を表示してくれます。また試しに **sshclient** から **sshserver** まででチェックしてみましょう。



さてここまでチェックしてきましたが、サンプルですし、ここで何も設定していないのにルートが通っているのには驚かなかったと思います。実の所、ここではノード毎に個別にルートが設定されているわけではなく、OSPF でやられているんですね。では OSPF でルートが自動生成されてる事を理解するために、少し弄てみましょう。

まず  を押してエミュレーションをストップさせましょう。シナリオでは n1 と n8 間のリンクがあるためすっかりとルーティングされていましたが、このリンクを右クリックから **Delete** でこのリンクにさよならしてみましょう。これで元のシナリオから変わった訳ですが、この状態でもきちんとルーティングされるでしょうか？ もう一度エミュレーションを開始して twonode ツールで確かめて見ましょう。



上手くいっているようですね。では最後にこのサンプルで動いているサービスの Ssh 接続を試したいと思います。ノードには初期パスワードが無いため、**sshserver** のターミナルを開いて *passwd* で設定し、このノードである事を示すためにファイルを作っておきましょう。

```
root@sshserver:/tmp/pycore.40663/sshserver.conf# passwd
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@sshserver:/tmp/pycore.40663/sshserver.conf# cd ~/
root@sshserver:/root# touch ThisisSSHServer
```

そして *sshclient* のターミナルを開いて *ssh* し、ファイルがあるかチェックしてみましょう。

```
root@sshclient:/tmp/pycore.40663/sshclient.conf# ssh 10.0.2.10
The authenticity of host '10.0.2.10 (10.0.2.10)' can't be established.
RSA key fingerprint is e4:d8:26:0b:70:56:53:4a:96:4b:f3:d0:89:24:98:a7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.2.10' (RSA) to the list of known hosts.
root@10.0.2.10's password:
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.2.0-67-generic i686)

* Documentation:  https://help.ubuntu.com/
```

```
6 packages can be updated.
6 updates are security updates.

New release '14.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

root@sshserver:~# ls
ThisisSSHServer
```

きちんと *sshserver* に接続できていましたね。

5 Pros/Cons

最後になりましたが、筆者が一年近く触った上で感じた CORE を使うと良い点と悪い点をここでは述べたいと思います。

Pros

- IOS イメージが必要になる GNS3 よりも使い始めやすく、ns-3 よりも触りやすい
- シミュレータではなく、エミュレータである事
- ホストのネットワークインターフェースをエミュレーション内に含めることができる
- ns-3 や ENAME との連携が可能
- 複雑なネットワークの場合、複数に分散してエミュレーションが行える
- Python でエミュレーションをコントロールできる

Cons

- エミュレーション内ではサービスは外部のソフトウェアを使うため、動かす場合には設定を書く必要がある^{*16}
- プロジェクトのゴールでも明示されているように、大規模なエミュレーション向けではない
- 知名度が極端に低いため、ドキュメントは公式のものと幾つかのブログしかない
- GUI がそんなにこなれてない、特に GNS3 と比べたら見た目の点では劣る^{*17}

^{*16} スクリプトが用意されている物もありますが、大半はそうではありません。

^{*17} 注意としてここでは **見たい目** にしか言及していません

6 おわりに

こんな感じで CORE について非常に短いですが解説できたので、多分満足してます。本当はワイアレスネットワークを解説したかったのですが、地獄が待っているので断念しました。この記事読んで、多少 CORE についてわかったと思いました？ 実はキャンパスは距離も表現している等の衝撃の事実があなたを待っているんですね。

それはともかく、家に大量にルーターを持ってる皆さんもそうでない皆さんも、こういったネットワークエミュレータと上手く付き合うとラボ無しでもテストできて良い話が発生します。自分も家では RouterBOARD や YAMAHA の赤いファイヤーウォールが動いている環境で CORE とか ns-3 を動かして自作プロトコルのテストをしています。ファイヤーウォールが赤いと目にも良いですね。

6.1 編集部員 (勝手に) 募集

プロトコルを書く人や Wireshark の Dissector を書く人いませんか？ あと最近 gRPC での http2 Dissector 書いてるんですが、gRPC に詳しい人いたら来て欲しいです。

Bibliography

- [1] Core manual — core 4.8 documentation. <https://downloads.pf.itd.nrl.navy.mil/docs/core/core-html/index.html>.
- [2] Open-source routing and network simulation. <http://www.brianlinkletter.com/>.
- [3] J. Ahrenholz, C. Danilov, T. R. Henderson, and J. H. Kim. Core: A real-time network emulator. In *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pages 1–7, Nov 2008.

SKK 紹介チラシ的な何か

文 編集部 ひだるま

1 SKK

SKK とは Emacs 上で動く日本語入力システムです。Emacs Lisp で実装されているので Emacs が動く環境ならどこでも動きます。以上。

2 Simple Kana to Kanji conversion program

「SKK 本体の開発は～」といった話は置いておいて、まあ皆 Emacs 上じゃなくても使いたいんで、色々な実装が色々な環境で走ります。どうして皆（大きい主語）使いたいかといえば、その変換システムが他とは違うからなんです。

先ず、変換したくない文字はそのまま打てます。一々変換候補が出て、**Enter**を押して、という必要がありません。直接入力→平仮名は**Ctrl-j**、平仮名⇄片仮名は**q**、平仮名（片仮名）→直接入力は**l**を叩くだけでモード変更ができます。平仮名（片仮名）から**Shift-l**（要は**L**）で全角入力になります。

肝心の変換ですが、**Shift**を押しながら 1 文字目を打ち始めて、単語を入れ終わった所で**Space**を押すと変換候補が表示されます。こう書くと少し間違いがあって、送り仮名がある場合、その文字で再び**Shift**を押しながら入力すると変換します。別の候補を出すには**Space**を押してください。前の候補に戻るには**x**を押します。

辞書登録が気軽にできる点も魅力の筈ですが、筆者は登録しなくても割と何とかなっているので説明しません^{*1}。

2.1 Ubuntu17.04

今だと fcitx-skk が導入が簡単です。何となく入れて、いつの間にか抜け出せなくなったのが筆者。

2.2 macOS X

Aqua SKK を使っている人が 2 人程知り合いに。

2.3 Windows 10

最近だと CorvusSKK がオススメらしいです。実際入れてみたらしっかり動いたので、良いと思います。

3 参考

- Openlab <http://openlab.ring.gr.jp/skk/index-j.html>
- DDSKK の紹介記事 <http://emacs.rubikitch.com/sd1408-ddskk/>

^{*1}できない。

もう FLOWERS の話しかしたくない

-京成バラ園へ行く-

文 編集部 linerlock

好きな作品をより深く知るためには何ができるだろうか。今のところ良い答えは見つけられていないが、登場人物が、あるいは作者が実際に目にした風景をこの目で眺めてみれば、何か違った視点を見つけられるかもしれない。聖地巡礼がしたい。京成バラ園に行きたい^{*1}。

...

京成バラ園は 1600 品種ものバラが咲き誇るローズ・ガーデンである。品種によって開花時期なども様々なので、四季折々のバラを楽しむことができるのが魅力である^{*2}。場所は東葉高速鉄道 八千代緑が丘駅から徒歩 15 分程度といったところで、残念ながらつくば市からは若干アクセスが悪い。車やバイクの場合は柏 IC を降りた後 1 時間程度下道をドライブする必要がある。高速道路からのアクセスこそ悪いが、その分園内は非常に静かである。広々とした園内はしっかりと作り込まれた庭園だけでなく、自然公園を思わせる芝や池などの休憩ができる場所も沢山あり、のんびりと過ごすことができる。利用客の多くはバラに時折目をやりつつ、ベンチなどで本を読みながら過ごす方が殆どであった。京成バラ園もゴールデン・ウィークなどのイベントシーズンには花まつりを開催して賑うが、静かさが最大の魅力なので、オフシーズンや平日などを利用して訪問することをすすめる。



^{*1} 筆者が実際に京成バラ園を訪れたのは 2016 年 8 月である。

^{*2} 開花時期については同施設のホームページで公開されているフラワー・カレンダーを参照するとよい。
<http://www.keiseirose.co.jp/garden/rosegarden/calendar/index.html>

京成バラ園のバラは素晴らしい, しかし今回の目的はそれだけではない. 綺麗なアーチを潜り, バラの間を通るブロムナードを抜け, 噴水広場を越えると目的のオブジェクトがその姿を見せる.



これはまさしく FLOWERS の登場人物たちが親交を深め, そしてなにより蘇芳君とマユリ君が相合傘を刻みつけたあの「東屋」だ. 聖アンブレカム学院 (FLOWERS の舞台) に登場する東屋は京成バラ園内のものがモデルになっていると言われている^{*3}. ちなみに, 「うみねこのなく頃に」に登場したベアトリーチェの幻想庭園のモデルでもある. 名称は「桂 由美 愛のローズガゼボ」と言うらしい. (gazebo: 西洋風東屋) ちなみにすぐ隣に設置されているプレートに目をやると...



「恋人の聖地」 こんなに絶対泣くでしょ. 蘇芳さん, マユリさん, 絶対に冬篇では幸せになってください.

^{*3} 独自研究. 実際にはデザインが大きく異なるのでガセネタの可能性もある.

ココアノノロイ

文 編集部 中原岬

アニメ『12 歳。』で提示された恋とは戦争であり、変化であり、愛情であった。ここで特筆すべきはココアちゃんの愛情が生み出してしまった迫害である。それは主人公・花日に向けられ、対立を引き起こし、結果として花日を変化させた。『12 歳。』にて対立項を演じたココアちゃんの、いや、ココアさんのお姿を忘れずにいよう。彼女は身を挺した。そして『12 歳。』という恋の物語を強化させた。つまり立役者なのだ、ココアさんは。愛憎というダブル・スタンダードへの、お家^{うち}での、お風呂での、お布団での、彼女の葛藤を想像し愛そう。幸いにもそれだけの美しさを製作側は提供してくれた。視聴者は彼女の存在により、世界が内包してしまっている矛盾に、そしてそれを起因とする正義の闘争に対峙した。

対立項への愛は、ときに裏切られる。映画『3 Idiots』はチャトル・ラーマリングラムというキャラクターを侮蔑し続け、ハッピーエンドという体で彼を虐め作品を締めてしまう。かつて宮沢賢治は「世界がぜんたい幸福にならないうちは個人の幸福はあり得ない」と記した*が、私はここまで過激になりえない。逆に、対立項の庄倒^{うしろ}が、みたい。そんな表現はあるのか。ある。白石晃士監督の『ノロイ』だ。

「みんな死んだ・・・」と投げかけられるメッセージに、われわれは不全を自覚する。白石監督の熟練・卓越したリアリティ撮影技法により映しだされる現象は、諦めを、潔癖に保持し続けた徳の唾棄を、迫る。描かれるのは我々の理を超えた死だ。「何でそんな言い方ができるんだよ」との提起、「お前やバイゾ」との注意、「もう全部ダメなんだよ」との理解が、死・死・死により突発する。そして霊体ミミズによるひたすらな否定が、否定の否定が、絶対的な虚無として新たな光芒をわれわれに差し込む。ココアでのそれを拡大して。



図：『ノロイ』のDVDパッケージ（後ろの光を発するのは『12 歳。』の視聴を共にしたF棟）

* 『農民芸術概論綱要』より

WORD 編集部への誘い

文 編集部 鳥

我々 WORD 編集部は情報科学類の学類誌「WORD」を発行している情報科学類公認の団体です。WORD は「情報科学」から「カレーの作り方」までを手広くカバーする総合的な学類誌です。年に数回発行しており、主に第三エリア 3A、3C 棟や図書館前で配布しています。

編集部の拘束時間には週一回の編集会議と、年に数回の赤入れや製本作業等発行に伴う作業があります。日常的に活動する必要はありませんので、他サークルの掛け持ちの障壁にはなりません。実際、多くの編集部員が他サークルと掛け持ちで在籍しています。

WORD 編集部には、情報科学類生や情報メディア創成学類生などが在籍しています。例えば、以下のよう
な人達が在籍しています。

natto 当たり屋

azuma962 APL プログラマの末裔

れつくす ホワイトハッカー日本代表

リザウド Lisp 大好きクラブ会員

びしょ〜じょ 百合仙人

下記に当てはまる方や、WORD に興味を持った方は是非、情報科学類学生ラウンジ隣の編集部室（3C212）へいつでも見学に来てください。時間を問わず常に開いています。

- AC 入試で入学した方
- それ以前に AC な方
- 印刷、組版や製版に興味がある方
- ネットワークの管理を経験したことがある方
- APL や Lisp が書ける方

その他質問がある方は、word@coins.tsukuba.ac.jp か、Twitter アカウントをお持ちの方は@word_tsukuba までお気軽にお問い合わせください。

情報科学類誌



From College of Information Science

メモリ 1枚、誰にでもあげる。号

発行者

情報科学類長

編集長

吉本 祐樹

制作・編集

筑波大学情報学群
情報科学類 WORD 編集部
(第三エリアC棟212号室)

2017年7月20日 初版第1刷発行 (512部)