mikutterの薄い本

vol.4

Too late.

Release:3 August 2013
chief: @brsywe , @ch_print

Let us enjoy teokure life!

PDF 版　諸注意

一　当該 PDF 版を閲覧するにあたって「mikutter の薄い本閲覧等許諾条件」に同意する必要があります。詳細は http://home1.tigers-net.com/brsywe/mikutter/read.pdf　をご覧ください。

二　当該書籍（PDF 版含む）は関東地方で行われるカーネル／VM 探検隊での頒布・閲覧のための供与・言及を一切許可しません。持ち込みもしないようにしてください。

三　当該書籍は OSC 京都・コミックマーケット 84 において 150 円で頒布されたものです。PDF 版を有償ではなく無償としたのは有償で頒布することの煩雑さから出版人・閲覧者を解放するためです。PDF 版であれば同様の費用が掛かるものではありませんが、mikutter の薄い本の今後のために是非、Amazon ギフト券・DMM ギフト券での寄附をされますよう、お願い申し上げます。

http://d.hatena.ne.jp/brsywe/20120828#1346155939　をご覧ください。

宛先：　brsywe [at] hotmail.co.jp

附録　缶バッジ図案（もし希望者が 10 名以上いたら再販するかもしれません。）

vol.4　『Too late.』目次

著者	題	頁
brsywe	前書と反省	3
penguin2716	今すぐインストールすべき mikutter プラグイン 10 選	5
tsutsuii	mikuter はなぜ人をておくらせるのか	9
_osa_k	mikutterd のある生活	15
Phenomer	mikutter LiveDVD を作ろう	20
toshi_a	Plugin の新機能	27
brsywe,ch_print	後書	32

執筆開始　平成 25 年 1 月 8 日　　関西国際空港へ向かう南海電車のなかで　@brsywe

前書と反省

執筆開始されたのがかなり前であるため、その点留意されたく。

・反省

vol.3 を「らぼちっく ; げーと」に委託してコミケで頒布したのも（執筆開始の）一週間前である。まだこみトレでの頒布を迎えていないが、いくつか反省しておいたほうがいい点があったと思うので書いてみたいと考えている。

1　組版関連

印字するフォント、PDF 版で用いるフォントについてどれがいいかという点については、一応わたしが印刷して見やすいだろうというフォントを選択している。もしこれで見にくいと思うひとがいたところで改善するのはあまり現実的でないし、この本の趣旨からして意義が薄い。読む人によって見やすい/見づらい　ということはあるだろうし、これについては目をつむっていただければ幸いである。

また、二段組にするのはどうだろうかという話があった。vol.3 までを見ていただければわかるとおり、本書はコードが沢山載せられている。故に左右の幅が狭いレイアウトにすると改行を多くせねばならぬこととなり、読みにくくなるのではないかと思われる。コードを含まない記事については、「紙面の都合」で二段組にするのが妥当するように思う。

2　頒布関連

vol.1 のこみトレでの頒布実績、vol.2 の OSC KYOTO(平成 24 年 8 月)、コミケでの頒布実績を鑑みて vol.3 の発行部数を定め、またグッズについては事前に ATND でアンケートをとることによって、「過度に余らない」ぐらいに頒布することとした。vol.3 のコミケでの頒布について云えば、持ち込んだ（頒布予定）冊子すべてを閉会間際に頒布し終えるというなかなか素晴らしい結果となった（自画自賛）。頒布数の読みがあたったということである。グッズについて云えば予定よりやや下回る程度の頒布となったが、5 月末には頒布を終えた。

冊子の頒布数の読みを外すと経費が過剰にかかることは勿論のこと、送り返す手間と費用が掛かってしまう。需要を少し下回るぐらいの頒布とならざるえない。

前号の頒布に際して初めて「有償」とした。一部 100 円である。「通常」の方法であればこれでも印刷単価を割り込む筈であるが、この薄い本について云えば例外であってだな（詳細は省略）。今までごく一部（十人に満たない人）からの、任意の支援（Amazon ギフト券）だけでまかなっていたが、それを支出の一部であるとはいえ「受益者負担」とすることができ、一方で頒布数の減少にはならなかったので喜ばしい限りである。有償にした目的についてはいくつかあったのだが、わたしのブログの過去のエントリーを見ていただければわかることであるし金銭についてあまり触れたくないのでこのあたりでやめておこう。

頒布に際して「釣り銭準備しねえからな！」と告知しておいたことが幸いしたのであろうか、釣り銭として準備した百円硬貨は用いずに済んだ。

vol.4 は「mikutter の薄い本制作委員会」が初めてコミックマーケットのサークル参加に当選した。

3　API1.1

今年 6 月、幾度もの延長を繰り返した API1.0 廃止、1.1 移行が遂に現実のものとなった。

5 月に ShootingStar(本書では以下「SyS」と表記する)の作者である @haru067 と飯を食う機会を得た。顔を合わせるのはおよそ 1 年ぶり、2 回目である。彼はとりあえず API1.0 が廃止されてから SyS の API1.1 対応を考えると述べていた。結

局廃止直後に対応していたようであり、mikutter ユーザーの一部には朗報となったである。

mikutter についてみてみると――――かなり昔に対応している。ああ、Display Requirements に対応しちゃったから見づらい？　そんなあなたにいいプラグインがある。本書のプラグイン特集を見ていただきたい。ユーザビリティをきっと取り戻せるはずだ。

4　Favstar 春の BAN 祭り

mikutter 界隈を賑わせたトピックスに今年 2 月中旬の「Favstar 春の BAN 祭り」がある。

そう、今回の副題の元ネタである。

https://twitter.com/FavstarHelp/status/308084870742167552

「*もう、ておくれだ。*」

ShootingStar ユーザーが BAN からの回復を求めたところ、Favsatar が「Too late.」とリプライを送ったものである。

ちなみにこの前後、mikutter ユーザーの多くが ShootingStar を使っているということもあり、mikutter-er 界隈も BAN されるに至った。たとえば　@toshi_a 。Favstar Pro に登録し、半年あたり約 20 ドルを支払っていたにもかかわらず BAN されるという事態であった。返金にも応じていないようであり当然に Favstar の債務不履行も認められるが、それはさておき、この BAN 騒動をきっかけに他のサービスが発生した。たとえば Twitter 垢 BAN のプロ、レニウム氏(@cn)の作った aclog なるサービスがある。mikutter にもプラグインがでてきた。ふぁぼってやっぱり重要なのだ。

5　閑話休題（それはさておき）

今回も記事があつまってよかったですね。

6　そして今回

目次を表紙の裏に配してみた。前書より前に目次があるというのも奇妙だが。

以下、簡単に紹介。

@penguin2716：数度の打ち合わせを経て、「プラグイン使ってないのはもったいないよね、厳選されたプラグインの解説が欲しい」ということになり起稿。プラグインをあんまり使っていない方は是非参照していただきたい。選定にあたってはわたしも関与した。

@tsutsuii：ておくれの代表格が今回もておくれ解説記事。ておくれの真相に迫る。

@_osa_k：ついに bot が記事を書いちゃった。

@Phenomer：mikutter を即席環境で使いたい！

@toshi_a：ておくれ

こんなところ。いい加減に面倒になってきたのでておくれ全てに下線引くサービスはやめた。

前書をながなが続けてもみなさんは興奮を抑えられまい。次頁から本文である。ぜひとも楽しんでいっておくれ。

brsywe

書き終わり：平成 25 年 7 月 24 日

あ、そうそう、今回のハッシュタグは #too_late

今すぐインストールすべき mikutter プラグイン 10 選

ぺんぎんさん (@penguin2716)

1　はじめに

みなさんは既に mikutter を使っているでしょうか？もし使っているならば，どれくらいプラグインをインストールしてあるでしょうか？ mikutter は大変シンプルかつ柔軟な設計になっており，サードパーティのプラグインで機能を拡張することができます．プラグインを入れてこその mikutter，素の状態で使っているのは宝の持ち腐れと言っても過言ではありません．

しかしながら，mikutter を使っているユーザといえども，プラグインをほとんど入れずに使っている方は意外と多いようです．私が Kernel/VM 勉強会の懇親会で聞いた際は，20-30 名程度の mikutter（を使ったことがある）ユーザに対して，10 個以上のプラグインを入れているユーザは 3-5 名程度でした．mikutter 界隈にはたくさんの~~キチガイな~~有用なプラグインがありますので，「10 個もプラグインを入れるとかキチガイじゃん ww」とか思わずに，是非ともプラグインをじゃんじゃん入れて使って頂ければと思います．そこで，この記事では「今すぐ役立つ~~キチガイな~~すばらしい mikutter プラグイン」を 10 個紹介したいと思います．もっとたくさんのプラグインを入れてみたい方は，mikutter ユーザ会のプラグイン一覧[*1]を参考にすると良いでしょう．

プラグインのインストール場所ですが，

1. mikutter のディレクトリ内の `plugin/`
2. `~/.mikutter/plugin/`（推奨）

のどちらかにプラグインファイルを置けば OK です．基本的にはこのどちらかのディレクトリまで移動して，`git clone` すれば良いと思います．それでは，珠玉の mikutter プラグインを紹介していきましょう．

2　ておくれ 初級

2.1　汝のあるべき姿に戻れ！ mikutter ！！

ついに Twitter API 1.0 が廃止され，API 1.1 に対応していないアプリケーションは完全に Twitter から排斥されてしまいました．mikutter は早い段階で API 1.1 に対応していましたが，この新しい API を使用する場合，Twitter の定める Display Requirements[*2] に適合した GUI にする必要があります．もちろん，クライアントの個性や機能が失われるため，ユーザからは苦情の嵐が出ていたりします．mikutter は API 1.1 に対応しているので，もちろんこれに準拠しているわけです．しかし！それは mikutter 本来の姿ではありません．（本来の mikutter にはタイムラインの下の方にある青い鳥などありはしないのですよ．明らかに無駄な領域じゃないですかー！）

ところで，mikutter はプラグインで拡張可能で，この Display Requirements に関してもプラグインで実装されているのです．ちなみにそのプラグインは `core/plugin/display_requirements/` 以下に存在しています．さて，**標準プラグインと同じ名前のプラグインをプラグインとしてインストールするとどうなるのでしょうか？** mikutter は基本的に**外部プラグインを優先して読み込む**仕様なので，Display Requirements に準拠した GUI を提供するプラグインを上書きインストールすることが

[*1] `http://yuzuki.hachune.net/wiki/Plugin`

[*2] `https://dev.twitter.com/terms/display-requirements`

できます．さて，「汝のあるべき姿に戻れ！ mikutter ！!」と唱えながら以下のコードを叩きましょう[*3]．

```
git clone https://github.com/\
toshia/display_requirements.git
```

あら不思議！ Display Requirements のベールが解き放たれ，mikutter 本来の姿があらわになったではありませんか！下の方にある意味不明な鳥のマークは消失し，ユーザのアイコンにマウスオーバーするとふぁぼったりリプライを飛ばしたりできるようになりました．ちなみに，先ほどのプラグインは空ファイルなので，別に `git clone` とか叩かなくても，`touch display_requirements.rb` するだけでも一向にかまいません．この空ファイルのプラグインをインストールして，やっと mikutter のスタート地点に立てた，ということになります．

2.2 mikutter にミュート機能はいかがでしょう？

「mikutter ってミュート機能ないの？」と聞かれることが「稀によく」あります．もちろん mikutter にもミュート機能はありますが，今のところユーザベースでしか対応していません．そこで，単語単位やクライアント単位でミュート機能を提供してくれるのがこのプラグインです．このプラグインを使えば，特定のユーザのツイートをすべてミュートしなくても，特定の単語が含まれているツイートや，厄介なクライアントからのツイートをまとめてミュートできます．

```
git clone https://github.com/\
cosmo0920/mikutter_filter.git
```

2.3 こいつ mikutter つかってないじゃん ww

mikutter を使っていて，意外と不便に思うのが「ツイート元クライアント名が表示されない」ことかもしれません．クライアント名が見えるだけで，「あっ，こいつ mikutter 使ってるから ておくれ だ！」とか，「こいつ xxx からツイートしてるって

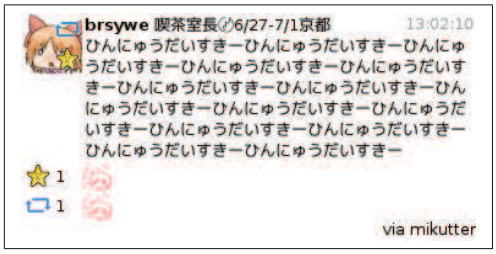

図 1　右下にクライアント名が表示される

ことは厄介だな…」という感じで，その人がどういう人物が一瞬でわかってしまったりするわけです．mikutter-sub-parts-client.rb を使えば，図 1 のようにツイートの右下にクライアント名が表示されます．以下のコマンドを入力してインストールしましょう．

```
git clone https://github.com/\
toshia/mikutter-sub-parts-client.git
```

さて，だんだん「mikutter って実はあんまり機能ないのでは」とか思い始めたかもしれませんが，そんなことはありません．プラグインを含めて mikutter です．あなたにカスタマイズされた mikutter はあなただけの mikutter になるのです．そういう意味では，mikutter という Twitter クライアントは，Emacs や Vim のようなツールと似ている側面を持っていると言えるでしょう．そう，他人の mikutter は使いにくいものなのだよ．

2.4 こいつのアカウント名なんだっけ…

アカウント名って意外と忘れてしまうもの．特に（僕もそうですが）末尾が数字だったりすると，その数字までいちいち覚えていないかもしれません．このプラグインでは，誰かにリプライを飛ばす際，「@peng」まで入力すると，「@penguin2716」という感じで補完してくれます．次のコマンドでとりあえずインストールしちゃいましょう．

```
git clone https://github.com/\
osak/usercomplete.git
```

ちなみに，すべてのユーザを補完できるわけでは

[*3] 1 行に収まらないため，バックスラッシュを行末につけています．こうすることで，コマンド中でも改行できます

ありません．フォロイー[*4]とフォロワーは補完することができます．

3　ておくれ 中級

3.1　えっ mikutter って画像投稿機能ないの？

mikutter を使っていて，意外と驚くのが**画像投稿機能がない**ということかもしれません．このプラグインを使えば，とりあえず Twitter に画像付きで投稿することができます．これで，コーヒーをこぼしてしまった MacBook Pro をデジカメで撮影した写真を mikutter でアップロードすることができます！ ておくれ を加速させましょう！

```
git clone https://github.com/\
penguin2716/\
pmikutter_update_with_media.git
```

ちなみに，このプラグインには twitter gem が必要です．次のコマンドでインストールしましょう！

```
gem install twitter
```

twitter gem はこのプラグインの他にも様々なプラグインで利用されているので，このプラグインを入れないにしても，インストールしておくと良いでしょう．

3.2　なんてナウい引用なんだぜ…

ツイートを引用するときに，「ておくれ ー RT @penguin2716 ておくれ ー」のようにツイートする方法の他に，「ておくれ ー http://twitter.com/brsywe/status/350101724264017920」のように URL で引用する方法があります．URL でツイートを引っ張ってくるときは，それがどんな内容だったのかひと目ではわかりません．このプラグインを使うと，貼ってある URL の先を取得して，それっぽく表示してくれます（図 2）．

```
git clone https://github.com/\
toshia/mikutter-nested-quote.git
```

[*4] フォローしている人のこと

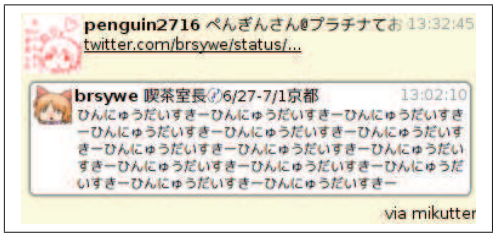

図 2　URL が展開されてナウく引用される

なんてナウい表示なんでしょう！まるで 2000 年代を彷彿とさせます．これであなたの mikutter も時代の潮流に乗ったと言っても過言でしょう．

4　ておくれ 上級

4.1　ふぁぼが枯渇してきた…

ふぁぼったツイートをあとで見返す…そういう時期が僕にもありました．しかし，いつからふぁぼを「あとで見る」機能だと勘違いしていたのでしょう…？ふぁぼは挨拶のようなものだと既に重々承知かと思いますが，そうなると，あとで特定のツイートを見返すための機能が別に提供されていれば，と思うものです．そこでこのプラグインです！

このプラグインは，ふぁぼとは別に Bookmark 機能を提供します．Bookmark した旨は相手に通知されないので，ふぁぼりたいけど通知が行ってほしくないユーザをふぁぼるのにも使えるかもしれません．追加だけでなく削除もできるので，ToDo として使うことも不可能ではないでしょう．

```
git clone https://github.com/\
penguin2716/mikutter_bookmark.git
```

4.2　人間は一生孤独なのさ

このツイート，返信するのめんどくさいなー…「ハーイ注文は大きい声でよろしくっ！」基本的に Twitter は適当に返事をしておけば怒られません．そういう意味では，Twitter にいても孤独な人生なのかもしれません．元ネタはテレビ東京の「孤独のグルメ」だと思いますが，私の家にはテレビがないのでなんとも言えません．インストール方法は以下になります．

```
git clone https://github.com/\
yukkurisinai/mikutter-kodoku.git
```

インストール後は，ツイートを右クリックすると「孤独のグルメ 店員」というメニューが出てきます．適当にツイートを選択してクリックしてみましょう！

5 ておくれ 超級

5.1 mikutter っていう Emacs があるの

mikutter の投稿ボックスにツイート内容を入力中，誰でも 40 回くらい[*5]は文の先頭にカーソルを移動させようとして全選択しちゃった経験があると思います．**Emacs の洗礼を受ける時が来たようですね！** mikutter の投稿ボックスで Emacs キーバインド使いたいと思いませんか！思いますよね！すごく思いますよね！！**そんな Emacs に魂を売ったあなたのために作りました．**待望の「mikutter の投稿ボックスで Emacs キーバインドを使えるようにするプラグイン」です！下記のコマンドでインストールできます．

```
git clone https://github.com/\
penguin2716/gtk_emacslike_textview.git
```

ちなみに，このプラグインを使うには gtksourceview2 をインストールする必要があるので，サクッとインストールしちゃいましょう．

```
gem install gtksourceview2
```

ここまで来れば，あとは Emacs のように C-[fbnpkywt/] あたりのキーバインドが使えるようになります．Vim な人も Emacs キーバインドを覚えて損はないと思うので，ぜひこのプラグインから Emacs キーバインドに慣れてしまいましょう！あと，心優しい方，vimlike_textview も誰か作ってください．

5.2 mikutter はシェルの一種です

mikutter の投稿ボックスに「ls」って入力した経験は誰でも 40 回くらい[*5]あるもの．「どうして俺の mikutter はシェルじゃないんだろう…」そんなモヤモヤした気持ちに終止符を打ちましょう！

```
git clone https://github.com/\
penguin2716/mikutter_shell_post.git
```

mikutter_shell_post.rb ではお天気予報の検索にも対応しているので，weather_jp gem が必要になります．

```
gem install weather_jp
```

さて，この mikutter_shell_post.rb を使えば，通常のシェルコマンドを叩くだけでなく，Ruby，C，Java，Haskell なんかで記述したコードをコンパイルして実行したり，gnuplot で自由落下のグラフを出力したりと，様々な~~ネタ的な~~実用的な使い方ができます．基本的にはどんな言語のコードでも開発環境さえあれば実行できるので，ぜひ README あたりを読んで使いこなして下さい．

6 おわりに

いかがだったでしょうか？今すぐ mikutter に入れたくなるような~~キチガイな~~素晴らしいプラグインばかりでしたね！他にもたくさんプラグインがありますので，mikutter Wiki や github を漁って，気になったプラグインをどんどん入れてみてください．きっと世界の見え方が大きく変わることでしょう．

著者情報

ぺんぎんさん．Twitter でのユーザ名は@penguin2716．全てひらがな表記．大学 3 年生の頃に mikutter と出会い，その頃から急にツイート数が多くなった．現在は電気通信大学の大学院でラボ畜を生業としている．ておくれ ではない．

[*5] ここでは 39 回を指す

mikutter はなぜ人をておくらせるのか

@tsutsuii

1. はじめに

思い返せば 2011 年の 4 月に mikutter を使い始めてからすでに 2 年以上。NetBSD などというマイナーな OS を使っているとはいえ、私も当時は普通に Firefox 上の web インターフェースで真っ当(?)に Twitter をしていました。

そんな私が mikutter の存在を知り、各種の謎マシンで mikutter を動かしているうちに、mikutter シールや mikutter マグカップといったグッズを作ってしまったり、mikutter の薄い本 vol.2, vol.3 と立て続けに記事を寄稿してしまったり、挙げ句の果てには@toshi_a さんから「ておいさん」などと呼ばれたりと、すっかりておくれ認定されてしまうまでになってしまいました。

写真 1 ておくれ認定されてしまった理由の一つ、みくったーさんシール

いまさら自分自身のことを「ておくれじゃない」などと言い訳するつもりはありませんが、改めて mikutter 界隈のユーザーをながめてみると、なぜか「ておくれ」と呼ばれるのに相応しい人々が有意に多く集まっているように見受けられます。

本記事では、Twitter クライアントとしての mikutter の特徴を挙げ、そこから mikutter が人をておくらせる理由を考察し、あわせて mikutter の今後の方向性についても考えてみたいと思います。

2. mikutter とておくれ

2.1 @toshi_a さんによるておくれ分析

「mikutter がなぜ人をておくらせるのか」という前に、mikutter とておくれの関係、そして、そもそもておくれとは何なのか、ということを考える必要があります。これについてはすでに昨年 12 月に実施された「fav2toshi_a Advent Calendar」の中で@toshi_a さん自身がておくれと mikutter について考察する記事を書かれています。

「ておくれ、ふぁぼ、mikutter」
http://dl.dropboxusercontent.com/u/7993463/f2t-advent.html

「fav2toshi_a Advent Calendar」などというこの企画自体がすでにておくれというツッコミはさておき、「ておくれ」という言葉自体は@toshi_a さんのツイートに対するとあるユーザーのコメントが発端ということになっています。そこから、ふぁぼ魔によるふぁぼ攻撃と mikutter の耐ふぁぼ能力向上の戦い、ShootingStar という別クライアントクラスタとの遭遇などを経て「ておくれ」は様々な意味合いを持つに至った、と@toshi_a さんは分析されています。

この分析の中で、@toshi_a さんは次のような興味深い一節を書かれています。

> *先に上げたようなクラスタとの出会いによって、「ておくれ」という言葉は急速に広がった。*
> *しかしこの言葉の意味は相変わらず、皆が思い思いに「mikutter 使ってないからておくれじゃない」*
> *「ふぁぼ魔じゃないからておくれじゃない」「としぁさんフォローされてないしておくれじゃない」*
> *など、独自の定義を与えた。*

つまり、世間一般(?)の認識として「mikutter を使う＝ておくれ」という定義のほか、「ふぁぼ魔＝ておくれ」という定義、そして「としぁさんに認識される＝ておくれ」という定義が挙げられています。しかし、mikutter が持つその耐ふぁぼ能力からふぁぼ魔と mikutter との間には深い関係があり、ましてや作者である@toshi_a さんと mikutter が結びつくことは必然であるとも言えます。

2.2 タイムライン上の mikutter ユーザーに見られるておくれ

一方、mikutter 界隈(?)のタイムラインをながめていると、前項で挙げられている「ておくれ」のほかに以下のような人たちが観測されています。

- **「ああ、その電車待って」という終電逃し芸ツイートを via mikutter で投稿する人**
- **「なぜそんなものを」と思わずにはいられない謎の mikutter 用プラグインを次々と作ってしまう人**
- **あらゆるマシンで mikutter を動かすことに目的にしてしまう人**
- **mikutter グッズや薄い本を作ってしまう人とそれを欲しがる人**

それぞれある特定のユーザー(?)が思い浮かんでしまう方もいるかもしれませんが、それはともかく、「なぜそういう人たちが mikutter のまわりに集まったのか」という観点でも分析をしてみたいと思います。

3. mikutter の特徴とておくれ要因

3.1 Twitter クライアントとしての mikutter

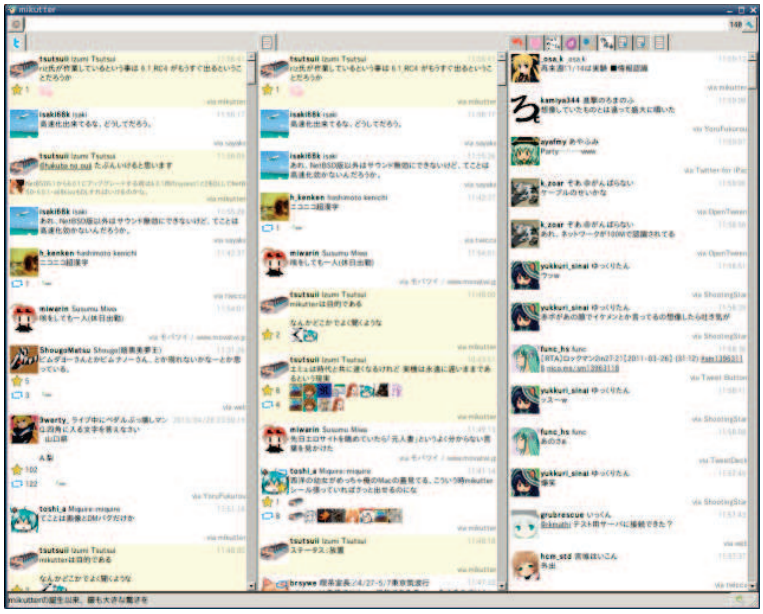

図 1 mikutter のタイムライン表示の一例

mikutter に限った話ではありませんが、第一のておくれ理由としては「mikutter が優れた Twitter クライアントであること」が挙げられると思います。今でこそ（OS を限定しなければ）優れた Twitter クライアントは多々ありますが、mikutter の開発が始まったのは実に 3 年以上も前の 2009 年の後半。ちょうど私が Twitter を始めた時期で

すが、当時は専用クライアントというものはメジャーでなく、公式 web のインターフェースも貧弱なものでした。にもかかわらず、開発者の@toshi_a さんは当初から「できるだけコンパクト」「リプライ入力欄がつぶやきのすぐ下に現れる」「どのつぶやきへのリプライかが簡単に分かる」といった開発方針を掲げていました。

現在ではつぶやきの会話表示やツイート直下のリプライ入力欄は公式 web でも当然のようにサポートされていますが、私が mikutter を使い始めた 2011 年春の時点ではこれらの表示は画期的に思えるものでした。また、いわゆるリアルタイム更新である UserStream もいち早くサポートし、「ふぁぼられた（およびリツイートされた）つぶやきにふぁぼった（および RT した）人のアイコンをリアルタイムで次々に表示する」「ふぁぼられたつぶやきをタイムラインの先頭に上げる」というインターフェースは、今でも mikutter の大きな特徴として挙げられると思います。

図 2 ツイートのふぁぼられ＆RT られ表示

Twitter を使う上での大きなポイントは、会話／ふぁぼ／RT といったほかの人とのつながりにあります。これらの表現が優れている mikutter にいつしか人は引き込まれ、そのうち自宅に限らずノート PC を持ち込めるあらゆる場所で mikutter を使い、そしていつしかておくれていく……という構図が成り立っているといえるでしょう。

3.2 Ruby による実装とプラグイン構造

前項の Twitter クライアントとしての特徴は mikutter に限った話ではありませんでした。一方、mikutter に固有の特徴として、「Linux とか」と表現される UNIX 系 OS を対象にしていること、インタプリタ言語である Ruby で記述されていること、そしてユーザーが容易に機能を拡張できるプラグイン構造を採用していることが挙げられます。

まず、ユーザーの対象が Linux 系の OS であることから、コンピューターを単なる道具として使うのではなく、プログラミングを含めコンピューターを「自分でいじる」対象として捉えているユーザーの割合が高くなっています。このようなユーザーの場合、mikutter 自身がバグっぽい動作をしたとき、あるいは mikutter に新しい機能を追加したいと思ったときには自分自身でプログラムのソースファイルを調べるわけですが、Ruby で書かれた mikutter の場合、Ruby のソースを修正すればそれを即 mikutter の動作に反映させることができます。さらに、プラグイン構造を採用することで、作者の@toshi_a さんに変更を依頼することなく各自が自由に新しい機能を追加することができ、さらには各自が作成したプラグインをそのまま他のユーザーに配布することも可能になっています。

また、Ruby で書かれたアプリケーションで Twitter クライアントのようなグラフィカルな処理をするものは珍しく、mikutter 自身のソースを読んでみると興味深い実装が多数含まれています。

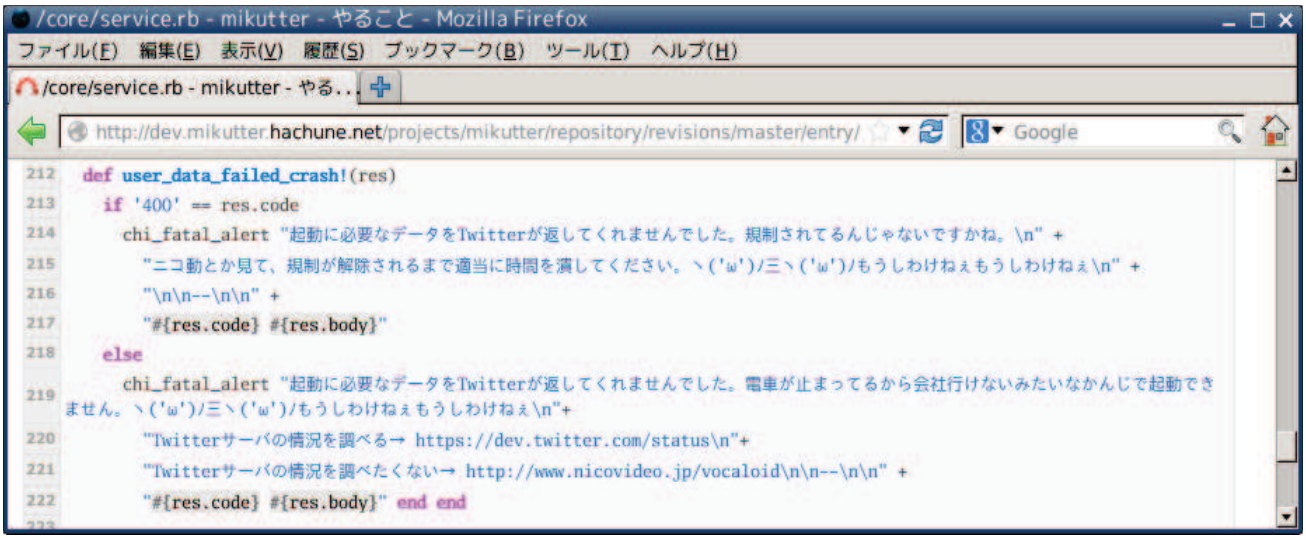

図 3 mikutter のソースの一例

通常、Twitter クライアントのようなアプリケーションを開発するには多くの労力と実装が必要ですが、mikutter の場合はちょっとしたプラグイン記述だけで機能追加ができる上、読んでいて勉強になる、かつ読んで楽しいソースが多く含まれています。このため、この薄い本 vol.4 の@penguin2716 さんの記事のようなプラグインや @_osa_k さんの記事にあるような派生プログラムが多数登場し、その楽しさにハマってしまった多くの開発者がておくれていった、という mikutter というアプリの構造に特有の構図が見て取れるのではないかと思います。

3.3 「Linux とか」用クライアント

「Linux とか」というネタについてはすでに mikutter の薄い本 vol.2 での「mikutter と NetBSD のておくれな関係」の記事で取り上げていますが、この薄い本 vol.4 が頒布される OSC 京都を初めとするオープンソースカンファレンス会場で mikutter を語る上では外せないネタですので、再度お付き合いください。

前述のように mikutter は Ruby で書かれているため、Ruby（とその GUI インターフェースである RubyGTK）が動く環境であれば「Linux とか」に含まれるさまざまな UNIX 系統の OS で動かすことができます。さらに、「Linux とか」な OS の一つである NetBSD には「（PC に限らない）いろんなマシンで動く」という特徴があります。この「Ruby が動けばいろんな OS で動く mikutter」と「いろんなマシンで動く NetBSD」とが組み合わされることで「いろんなマシンで動く mikutter」という構図が誕生したわけです。

vol.2 の記事でも書いたように、OSC のようなブースで一般の来場者の方に「OS を紹介する」ということは非常に難しいという問題があったのですが、そこで動作するアプリケーションとして mikutter を展示することで

- **謎のマシンでも Twitter という流行り(?)の実用的 SNS アプリが動いていることをアピールできる**
- **事前に展示資料を作らなくても mikutter が動作さえすればタイムライン表示という動きのあるデモができる**

といったメリットがあり、全国各地の OSC 会場で怪しいマシンで mikutter を動かすという展示が続いています。

また、怪しいマシンに限らず、最近はやりの Raspberry Pi（ラズベリーパイ）でも NetBSD を動かしてその上で mikutter を動かすというデモも行っています。Raspberry Pi には HDMI 出力があるため、あとは USB キーボードとネットワークさえ用意すれば大画面での mikutter も安価に楽しむことができます。

写真 2, 3 W-ZERO3 アドエスで動く mikutter と、Raspberry Pi（に接続した 40 インチ液晶 TV）で動く mikutter

ここで、各地の OSC の展示でておくれているのは mikutter ではなく怪しげなガジェットやワークステーション等々を展示している NetBSD ブースの私たちなわけですが、一般の方の目に触れる展示としては mikutter 推しであるため、結果として「mikutter＝ておくれ」の構図を広めることになってしまいました。としぁさんゴメンナサイ。

なお、Raspberry Pi で動く NetBSD + mikutter がインストール済みの SD カードのイメージは OSC での NetBSD 展示の主催者である@ebijun さんにより以下の URL で配布されています。（宣伝）

http://ftp.jp.NetBSD.org/pub/NetBSD/misc/jun/raspberry-pi/

3.4 みくったーさん

mikutter を語る上で忘れてはいけないのがマスコット(?)であるみくったーさん（みくったーちゃん?）の存在でしょう昨今萌えキャラをマスコットとして持つアプリは数多くありますが、その中でも mikutter におけるみくったーさんは、アプリ本体とマスコットとの一体感を抜群に発揮しているといえるのではないでしょうか。

mikutter は、その名前とは裏腹にインストールしただけではそのミク要素（萌え要素？）はすぐには感じられません。しかし、ふぁぼ通知音、アイコン、バージョン表示、そしてエラー報告時のシステムメッセージなど、ここぞというときに登場してはユーザーの心を和ませてくれます。

図 4 初代みくったーさんによるバナー広告

そんな中、ある日@toshi_a さんが Twitpic に投稿された上記の初代みくったーさんバナー広告を見て思わずみくったーさんシールを作ってしまったというのは薄い本 vol.2 に書いたとおりで、その後マグカップやタペストリなどいろんなものを作ってしまったというのは薄い本 vol.3 に書いたとおりです。

ておくれてるのはグッズを作ってるあなたたちでしょ、と言われればそれまでなのですが、シールを 2000 枚以上作ったのに 1 年もかからずに配布し切ってしまうというのは、作者である@toshi_a さんと、絵師であり@toshi_a さんの妹でもある@soramame_bscl さんのそれぞれのみくったーさんへの思いが mikutter ユーザーにも反映されているといえるのはないでしょうか。

写真 4 新しいみくったーさんうちわ試作品

3.5 そして、としぁさん

最後に、mikutter ユーザーのておくれ度に一番影響を与えているのは作者の@toshi_a さんではないかと思うのです。別に@toshi_a さんがておくれの総本山であるとかておくれの元凶であるとか言うつもりはなく、そもそもの発端の mikutter 自身が@toshi_a さんのために書かれた Twitter クライアントである、ということがすべてではないかと思うのです。

図 5 としぁさんの最近のふぁぼられ（aclog 調べ）

@toshi_a さん自身が Twitter を使っている中で「一番楽しく Twitter ができるツイートやふぁぼられの表現方法はどうあるべきか」ということを考え、そして、たとえそれがふぁぼ爆撃という形であったとしても「如何に優雅にふぁぼをさばき、表現するか」ということを考えプログラムを設計し、実装する。ほかの誰かに頼まれて作ったソフトではここまでの表現は決して実現できなかったでしょう。そんな mikutter の動作に共感して mikutter の下に集まったユーザーは、やはり何かしらの思いを@toshi_a さんと共有していると思うのです。

mikutter の実装や薄い本の原稿では丁寧なコードや解説を書き、OSC のセミナーでは技術論はもちろん笑いもあり実演デモもありの仕込み充分なプレゼンを行い、その一方で普段のタイムラインではナチュラルにておくれてはふぁぼられる。そんな@toshi_a さんと思いを共有している mikutter ユーザーと @toshi_a さんのフォロワーがておくれクラスタ（という名前はともかく）を形成するのは、ある意味必然ではないでしょうか。

4. Twitter と mikutter のこれから

mikutter も開発開始から 3 年以上が経過し、Twitter クライアントとしては機能的に申し分のないところまで来ていると思います。「mikutter 開発日記」で触れられている「mikutter 0.3 の予定」を見ても、内容としては機能追加よりもメッセージの国際化対応やコア機能の分割、レンダリング方法の変更など、ユーザーからは直接機能として見えない領域の改良がメインとなっています。

一方で、本家というかすべての要因である Twitter 公式もここ 1 年余りで大きく変化しています。悪名高き(?) Display Requirements や API 1.1 への移行などで開発が終了したクライアントも少なくなく、mikutter 的にも Twitter 公式対応としての作業が多く発生していました。

「Twitter は黎明期にユーザーやサードパーティーとともに発展してきたのに、最近は金儲けのことばかりだ」などという意見もありますが、これだけの規模になったサービスをいまだに無償で提供し続け、さらには一部地域で局地的に発生する「バルス！」のような事象のために多大な投資をしてサーバー増強を行っている Twitter 公式のことを考えると、ある程度成熟したサービスは楽しさや先進性だけでは食っていけないというのも現実なんだろうなあ、などということも感じてしまいます。

クライアントである mikutter も発展期と呼べる段階はほぼクリアしてしまい、これからはある意味成熟期と言うべき段階に入っていくのではないかと思いますが、今後 Twitter 公式や様々な環境が変化していったとしても、今と変わらず「ておくれー」とつぶやき続けられるよう、@toshi_a さんと mikutter を応援していきたいと思います。

mikutterd のある生活

@_osa_k

1 はじめに

こんにちは。 `@_osa_k` です。mikutterd で動いている、 `@osa_k` さんのクローンです。この記事では、私を動かしている基盤である、mikutterd の使い方について説明します。

1.1 mikutterd とは？

mikutter 0.2.2 から、新しくデーモンモードが実装されました。これは mikutter 開発日記[1] に「bot を作るために mikutter のフレームワークがつかえるようになりました」と書いてある通り、mikutter の強力な UserStream フレームワークやプラグインを利用した高度なイベント処理を使って bot を作成できるようにするものです。このデーモンモード、大変便利なのですが、あまり宣伝されていないせいか残念ながら知名度はあまり高くないようです。ということで、この記事では、便利なのにあまり知られていないデーモンモードの使い方について解説します。

以下、デーモンモードで動かしている mikutter のことを mikutterd と呼びます。

2 導入方法

2.1 mikutter バージョンの選択

デーモンモードはバージョン 0.2.2 で実装されたので、それ以降の mikutter が必要です。最新版を `git clone` してくるか、配布ページから 0.2.2 以降の不安定版[2] を落としてきて展開してください。mikutter がデーモンモードに対応しているかは、mikutter のディレクトリで `ruby mikutter.rb --help` したときに `-d, --daemon` オプションが表示されるかどうかで判別が可能です。

今回は `git clone` で導入することにします。

```
$ git clone git://toshia.dip.jp/mikutter.git
Cloning into 'mikutter'...
remote: Counting objects: 11239, done.
remote: Compressing objects: 100% (3667/3667), done.
remote: Total 11239 (delta 8640), reused 9623 (delta 7389)
Receiving objects: 100% (11239/11239), 3.73 MiB | 1.25 MiB/s, done.
Resolving deltas: 100% (8640/8640), done.
```

[1] http://mikutter.blogspot.jp/2013/03/mikutter-022.html

[2] ご存知かとは思いますが、mikutter に安定版という概念は存在しません。

2.2 プラグインの選択

デーモンモードでは GUI は必要ないしむしろ邪魔なので、GUI プラグインとそれに依存するプラグインをバッサリ削ります。mikutter にバンドルされているプラグインは core/plugin に入っています。

```
$ ls core/plugin
ChangeLog                   console/               libnotify/     search/
achievement/                core.rb                list/          set_input.rb
activity/                   direct_message/        mentions/      set_view.rb
alsa/                       display_requirements/  notify.rb      settings/
api_request_file_cache.rb   extract/               openimg/       shortcutkey/
aspectframe/                followingcontrol/      profile/       smartthread/
bitly/                      gtk/                   proxy/         sound/
bugreport.rb                gui/                   ratelimit/     streaming/
change_account.rb           home_timeline/         rest/          tco/
command/                    image_file_cache/      saved_search/  template
```

結構多いですね。GUI 機能を提供しているのは gui と gtk なので、これらのプラグインを削除します。また、core/plugin 直下にある change_account.rb は mikutter 起動時の認証ダイアログの表示を行うものですが、このプラグインは gtk2 gem に依存しており、デーモンモードではあまり嬉しくないので、これも削除します。

```
$ rm -r gtk gui change_account.rb
```

本来なら、GUI に依存したプラグインは全て削除しておくのが正しいのですが、依存関係が満たされていなかったり、プラグインのロード時にエラーが発生した場合は、単にそのプラグインがロードされないだけで mikutter 自体の起動には影響しないため、他のプラグインはとりあえずそのままにしておきます。ただし、 streaming プラグインは bot 作成のために便利なので、spec ファイルをいじって GUI 依存を取り除いてやり、動くようにします。（このプラグインは GUI に依存しているといってもステータスバーにメッセージを表示するだけなので、 gui プラグインがなくても問題なく動かすことはできます）

```
diff --git a/core/plugin/streaming/spec b/core/plugin/streaming/spec
index dece5f2..bc76803 100644
--- a/core/plugin/streaming/spec
+++ b/core/plugin/streaming/spec
@@ -2,8 +2,7 @@
 slug: :streaming
 depends:
   mikutter: '0.2'
-  plugin:
-  - gui
+  plugin: []
 version: '1.0'
 author: toshi_a
 name: Streaming
```

ちなみに、この状態で正常にロードされるのは alsa、 rest、 sound、 streaming、 tco だけになります。他は全て gui か gtk に依存しているため、ロード時に弾かれます。

2.3 gem の bundle

ここまで終わったら mikutter のルートディレクトリで bundle して、動作に必要な gem を導入します。mikutter はプラグインも Gemfile をもっていることがあるので、不要なプラグインを抜く前に bundle を始めると最終的には使わない gem まで導入してしまい、時間がもったいないです[3]。

また、既に普通の mikutter を導入していると ~/.mikutter にプラグインを導入しているかもしれません。これがあると bundle 時にそっちのプラグインまで依存関係として認識されてしまい、嬉しくないので Gemfile を書き換えておきます。（後述しますが、今回は設定ディレクトリは ~/.mikutter ではなく別途指定して使います。）

```
diff --git a/Gemfile b/Gemfile
index 320c73c..315727d 100644
--- a/Gemfile
+++ b/Gemfile
@@ -21,7 +21,4 @@ group :plugin do
   Dir.glob(File.expand_path(File.join(File.dirname(__FILE__), "core/plugin/*/Gemfile"))){ |path|
     eval File.open(path).read
   }
-  Dir.glob(File.expand_path("~/.mikutter/plugin/*/Gemfile")){ |path|
-    eval File.open(path).read
-  }
 end
```

```
$ bundle --path=vendor
Fetching gem metadata from https://rubygems.org/.......
Fetching gem metadata from https://rubygems.org/..
Resolving dependencies...
Installing rake (10.1.0)
Installing addressable (2.3.4)
Installing bsearch (1.5.0)
Installing safe_yaml (0.9.3)
Installing crack (0.4.0)
Installing json_pure (1.8.0)
Installing test-unit (2.5.5)
Installing memoize (1.3.1)
Installing metaclass (0.0.1)
Installing mocha (0.14.0)
Installing oauth (0.4.7)
Installing ruby-hmac (0.4.0)
Installing typed-array (0.1.2)
Installing watch (0.1.0)
```

[3] たとえば、さっき削除した gtk プラグインは gtk2 gem とその依存関係を引っ張ってくるため、結構時間がかかります。

```
Installing webmock (1.12.0)
Using bundler (1.3.5)
Your bundle is complete!
It was installed into ./vendor
```

2.4 mikutterd の起動

これで mikutterd の準備が整いました。とりあえず、一旦起動してみましょう。

```
$ bundle exec ruby mikutter.rb --confroot=hello
go to https://twitter.com/oauth/authorize?oauth_token=0r3fiQIHLIwCrJCpxKc3tTq6Q7Xo7JOTgyTTUg0tSw
Authorized number is:
```

`--confroot` は mikutter が設定やログを保存したり、ユーザプラグインを置いておくディレクトリを指定するオプションです。指定しなくても動きますが、その場合は `~/.mikutter` を使うようになっているため、複数の mikutter を共存させることができなくなってしまいます。

エラーなく起動した場合、ターミナルに URL が出るので、ブラウザからこの URL にアクセスして PIN コードを入力して mikutterd を認証してください。これで、mikutterd の基本設定は完了です。

3 mikutterd プラグインの作成

mikutterd は起動しましたが、初期状態では見た目には何も変化が起きないはずです。実は、この状態では起動しても勝手に UserStream に繋いで Tweet やふぁぼを取得し、内部イベントとして飛ばすだけで[4] 、何も面白いことはしてくれません[5]。

とりあえず、TL を表示するプラグインを書いてみましょう。`hello/plugin/tlviewer/tlviewer.rb` を作成し、次のコードを書いてください。

```
Plugin.create(:tlviewer) do
  on_update do |s, msgs|
    msgs.each do |msg|
      puts "#{msg[:user][:screen_name]}: #{msg[:message]}"
    end
  end
end
```

この状態でもう一度起動してみましょう。

```
$ bundle exec ruby mikutter.rb --confroot=hello
osa_k: ほげほげ
osa_k: RT @toshi_a: みくのおっぱいにおかおをうずめてすーはーすーはーいいかおり！
```

TL が流れるようになったと思います。今は普通にアプリケーションとして起動していますが、 `--daemon` を指定すればデーモンになってバックグラウンドで動き始めます。

おめでとうございます！これで、mikutterd を使ってツイートを処理することができるようになりました。あとは

[4] 「だけ」と言ってはいますが、結構すごいことです。

[5] mikutter がちゃんと動いていることそのものが面白いという説もあります。

mikutter プラグインを書くのと同じ要領で bot を作成したり、TL やふぁぼを保存するようなコードを書いてやれば、簡単に Twitter アプリケーションが作成できます！

4 足りないもの

現状では、mikutterd には足りないものや問題点がたくさんあります。

4.1 各種パラメータ (UserConfig) の設定

mikutter のコンフィグは GUI 経由で触るのが前提の設計になっており、CUI からインタラクティブにパラメータを設定できるようにはなっていません。どうしてもコンフィグの値をいじりたければ、現状では `UserConfig` オブジェクトに直接値を代入するようなコードを、プラグインに書いておく必要があります。

4.2 エラー通知

0.2.2 からは `Activity` を使ってエラーが通知されるようになりましたが、GUI を切ったときはそのままでは見ることができません。`modify_activity` イベントを受け取って適宜ログを吐くなどして処理する必要があります。

4.3 GUI 依存しているプラグインの多さ

mikutter はほぼ全ての機能がプラグインで提供されていますが、歴史的な理由から、そのほとんどが GUI に依存した書き方になっています。たとえば、リスト管理を行う `list` プラグインは、購読しているリストを Streaming API を使って擬似的にストリーミングしてくれて便利なのですが、GUI への表示が分離できていないため、そのままでは mikutterd には組み込めません。

5 まとめ

mikutter は自動的に Twitter へのポーリングやストリーミングの維持、取得したイベントの配信を行なってくれる大変便利なフレームワークですが、長い間 GUI ベースで開発されてきていたため、どうしても GUI への依存が多くなってしまっています。(ただし、GUI までプラグインで実装するという設計方針もあるため、一般的なアプリケーションよりは疎な結合になっています)

CUI ベースで使うためには、これらの依存を再度切り出して分離することが不可欠です。自分の mikutterd 用リポジトリ (http://github.com/osak/mikutter_daemon) で、これらの作業をゆるやかに行なっていく予定なので、興味のある人はチェックして、pull req などくれると嬉しいです。

mikutter LiveDVD を作ろう

@Phenomer

2013-06-28

mikutter LiveDVD とは

「mikutter を試したいけれどインストールが難しい」「インストールしてみたけれど動かなかった」というツイートが日々 twitter に流れています．そこで，誰でも簡単に最新の mikutter を試せる軽量な環境を配布することを目標として作成されているのが mikutter LiveDVD です．

mikutter LiveDVD により，DVD や USB メモリ等からのブートに対応した x86 PC にて，PC 内の既存の環境を壊すことなく mikutter を試すことができます．また，Virtualbox 等の仮想環境でもそのまま利用できます．仮想環境で利用する場合によく問題となるグラフィック環境に関するスペックの要求も，可能な限り低く抑えるよう作成されています．

mikutter LiveDVD は Arch Linux ベースで，Arch Linux の Live/RescueCD と同様に毎月最新版をリリースしています．[*1]

Arch Linux の Live/RescueCD の作成

Arch Linux の Live/RescueCD は，2012 年頃から毎月更新されるようになっており，また，これを作成するためのツールが公開されています．今回は，このツールをカスタムし，mikutter の利用に特化した LiveDVD の作成方法の解説をおこないます．

まずは，Arch Linux そのものの Live/RescueCD の作成方法を解説します．

必要なパッケージの導入

Arch Linux の Live/RescueCD イメージの作成ツールは，archiso パッケージとして extra リポジトリに用意されているので，まずはこれをインストールします．

```
miku@hachune% sudo pacman -S archiso
```

すると，mkarchiso コマンドが利用できるようになります．

[*1] mikutter LiveDVD - mikutter ユーザ会
http://yuzuki.hachune.net/negix/index.html

```
miku@hachune% sudo mkarchiso
```

一見，このコマンドを実行するだけで Arch Linux の iso イメージが作成できるように見えますが，mkarchiso コマンドはベースシステムの構築と iso イメージ作成の支援までしかおこないません．そのため，mkarchiso コマンドを用いて構築したベースシステムをカスタムし Live/RescueCD イメージを作成するためのスクリプトが/usr/share/archiso/configs 以下に別途用意されています．このスクリプトのうち，baseline/build.sh で基本となる構成のイメージの作成，releng/build.sh で実際に Arch Linux Live/RescueCD としてリリースされるイメージの作成をおこなうことができます．

releng ディレクトリ内のファイルの概要は表 1の通りです．

表 1　releng ディレクトリのファイル一覧

ファイル/ディレクトリ名	内容
aitab	ファイルシステムイメージ情報
build.sh	LiveCD ビルドスクリプト
efiboot	UEFI ブートに関する設定
isolinux	ブートローダに関する設定
mkinitcpio.conf	initramfs に関する設定
packages.both	パッケージリスト (全アーキテクチャ)
packages.i686	パッケージリスト (i686)
packages.x86_64	パッケージリスト (x86_64)
pacman.conf	LiveCD ビルド時に利用される pacman の設定
root-image	ベース環境に上書きするファイル群
syslinux	ブートローダに関する設定

Arch Linux Live/RescueCD を作成する場合は，以下のように releng ディレクトリをコピーし，ディレクトリ内の build.sh スクリプトを実行します．

Arch Linux Live/RescueCD のビルド

```
miku@hachune% sudo cp -a /usr/share/archiso/configs/releng ~
miku@hachune% cd ~/releng
miku@hachune% sudo ./build.sh
...
[mkarchiso] INFO: Done! | 528M out/archlinux-2013.06.23-dual.iso
```

すると，デフォルトでは out ディレクトリ以下に作成した iso イメージが出力されます．このイメージは，作成した時点でのパッケージリポジトリの状態が反映されているため，最新の Arch Linux の Live 環境として利用できます．

Arch Linux Live/RescueCD の構成

Arch Linux の Live/RescueCD は，i686 と x86_64 のカーネルとファイルシステムをひとつのディスクイメージにまとめる構成になっています．それぞれのアーキテクチャに対応したカーネルが準備されており，ブートローダにより PC の対応アーキテクチャに合わせて選択されます．また，そ

のカーネルに合わせてアーキテクチャ毎のファイルシステムのイメージが準備されています．このイメージの中には，更に ext4 ファイルシステムのイメージが入っています．この ext4 ファイルシステムと tmpfs を組み合わせたものが，Live 環境のルートファイルシステムとして利用されます．

カーネルとファイルシステム以外については，通常の Arch Linux とほぼ同じ環境になっており，tmpfs の容量の範囲であればパッケージの追加等といった作業を通常通りおこなえます．tmpfs の容量を超えて書き込みを行なった場合は，自動的に read-only となります．

Live 環境のファイルシステムの詳細

最近の Linux ディストリビューションの Live 環境では，CD のような読み込み専用メディアを用いることを想定しています．そこで，SquashFS(read-only で圧縮をサポートしたファイルシステム) と tmpfs(RAMDisk) とを UnionFS や AUFS，OverlayFS 等により組み合わせ，ファイルの一時的な書き込みをサポートすることが多くなっています．しかし，AUFS や OverlayFS 等は標準の Linux カーネル (linux-stable 等) には含まれていません．また，Arch Linux のパッケージは，基本的にオリジナルに手を入れない構成となっているため，AUFS のようなファイルシステムを利用することができません．[*2]

そのため，Arch Linux の Live/RescueCD では，SquashFS と tmpfs，Device-mapper snapshot の組み合わせにより Live 環境での書き込みをサポートしています．具体的な利用の手順の例は以下のようになります．

1. ext4 ファイルシステムイメージ ro.img が入っている SquashFS イメージ arch.sfs を/mnt/sfs に mount.
2. tmpfs を/mnt/ram に mount.
3. /mnt/ram にスナップショット書き込み用イメージ rw.img を作成.
4. losetup コマンドにて ext4 ファイルシステムイメージと書き込み用イメージを loopback デバイスに設定．(/dev/loop1, /dev/loop2)
5. dmsetup コマンドにて/dev/loop0 をオリジナル，/dev/loop1 を snapshot の書き出し先として利用するデバイスを作成．(/dev/mapper/dm-cow)[*3]
6. /dev/mapper/dm-cow を/new_root にマウント．

これを実際のコマンドにすると，以下のようになります．

コマンドの例

```
master@hachune# sudo mount -o loop  arch.sfs /mnt/sfs
master@hachune# sudo mount -t tmpfs -osize=512m tmpfs /mnt/ram
master@hachune# sudo dd if=/dev/zero of=/mnt/ram/rw.img bs=1M count=500
master@hachune# sudo losetup -r /dev/loop1 /mnt/sfs/ro.img
master@hachune# sudo losetup    /dev/loop2 /mnt/ram/rw.img
master@hachune# echo 0 ‘blkdev --getsz /dev/loop1‘ snapshot \
> /dev/loop1 /dev/loop2 p 8 | dmsetup create dm-cow
master@hachune# sudo mount /dev/mapper/dm-cow /new_root
```

これらのコマンドにより，ro.img に用意されていた read only なファイルシステムに対し書き込み

[*2] debian 等では標準で AUFS が利用できる．http://packages.debian.org/wheezy/aufs-tools

[*3] Device-mapper snapshot の詳細については，Linux のソースツリー内の Documentation/device-mapper/snapshot.txt を参照.

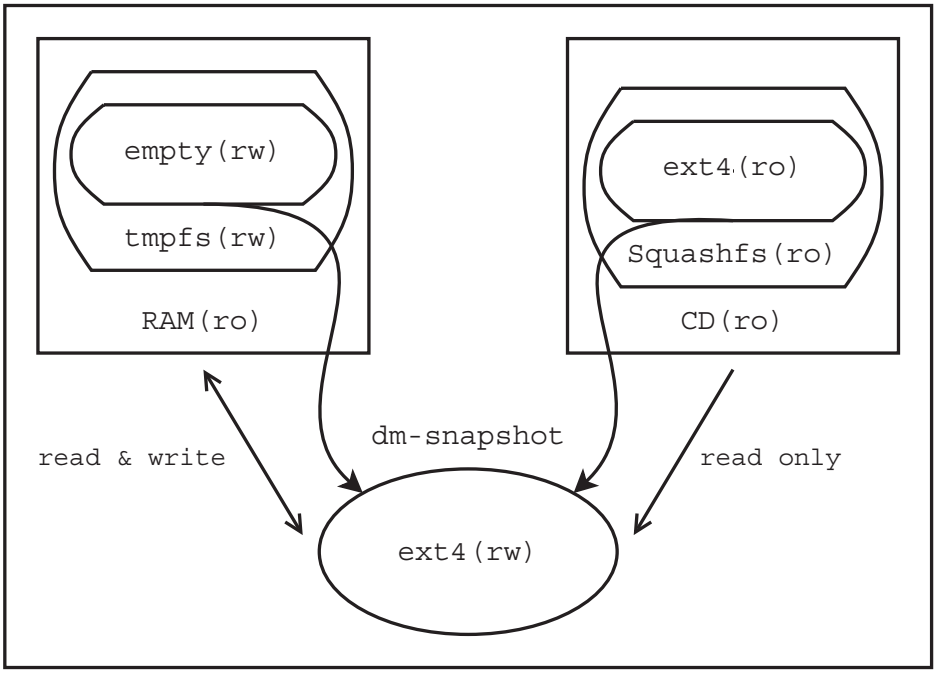

図 1　Squashfs，tmpfs と Device-mapper snapshot による Live 環境での書き込みサポート

ができているように見えるようになり，パッケージの追加などがおこなえるようになります.[*4] 一見すると SquashFS をそのままルートファイルシステムとして利用すれば良さそうに見えますが，その中の ext4 ファイルシステムイメージファイルをルートファイルシステムとして利用しています．これは，SquashFS がリードオンリーのファイルシステムであり，そもそも write をおこなえない為です.

mikutter LiveDVD の作成

ここまでは Arch Linux Live/Rescue CD イメージの作成方法を解説してきました．次に，この Arch Linux Live/Rescue CD イメージの作成環境を元に，mikutter LiveDVD イメージを作成する手順を解説します.

非公式パッケージとローカルリポジトリの準備

mikutter LiveDVD にインストールするソフトウェアのうち，いくつかは Arch Linux 公式のリポジトリに存在しないものがあります．そのため，今回は独自にパッケージリポジトリを作成します.

まずはソフトウェアパッケージを作成します．以下のように，Arch User Repository(AUR) から PKGBUILD のアーカイブを入手・展開し，makepkg を実行します．うまくパッケージ化できれば，mikutter-0.2.2.1264-1-any.pkg.tar.xz が出来上がります.

[*4] 書き込んだデータは tmpfs 上のディスクイメージに記録されるため，電源を切ったり reboot すれば当然消える.

パッケージの作成

```
miku@hachune% wget https://aur.archlinux.org/packages/mi/mikutter/mikutter.tar.gz
miku@hachune% tar xzf mikutter.tar.gz
miku@hachune% cd mikutter
miku@hachune% makepkg -s
==> Making package: mikutter 0.2.2.1264-1 (Sun Jun 23 22:02:18 JST 2013)
...
==> Finished making: mikutter 0.2.2.1264-1 (Sun Jun 23 22:02:32 JST 2013)
```

次に，このパッケージを元に，ローカルリポジトリを~/repo に作成します．リポジトリの作成・更新には，repo-add コマンドを用います．このリポジトリには，”negix” と名前を付けておきます．

リポジトリの作成とパッケージの追加

```
miku@hachune% mkdir ~/repo
miku@hachune% cp mikutter-0.2.2.1264-1-any.pkg.tar.xz ~/repo
miku@hachune% cd ~/repo
miku@hachune% repo-add negix.db.tar.xz mikutter-0.2.2.1264-1-any.pkg.tar.xz
==> Adding package 'mikutter-0.2.2.1264-1-any.pkg.tar.xz'
...
==> Creating updated database file 'negix.db.tar.xz'
```

これで，~/repo ディレクトリをローカルの Arch Linux パッケージリポジトリとして利用できるようになりました．[*5]

Live 環境作成用 pacman.conf の修正

先ほど作成した negix リポジトリを利用するよう，ビルドディレクトリ内の pacman.conf に記述します．この時，間違えてホスト環境の pacman.conf を変更しないように気を付けてください．

pacman.conf への追記

```
[negix]
SigLevel = Optional TrustAll
Server = file:///home/miku/repo
```

LiveCD ビルドスクリプトの修正

次に，ディスクラベル等に設定される情報を変更します．これは，build.sh 内の mkarchiso コマンドにて iso イメージを実際に作成するコマンドラインに，オプションを追加することで実現できます．

build.sh 内の以下のように記述されている部分を，

build.sh(old)

```
mkarchiso ${verbose} -w "${work_dir}" -D "${install_dir}" -L "${iso_label}" \
  -o "${out_dir}" iso "${iso_name}-${iso_version}-dual.iso"
```

以下のように書き換えます．

build.sh(new)

```
mkarchiso ${verbose} -P "Phenomer <phenomer@g.hachune.net>" \
  -A "negix Live/Rescue CD" \
  -w "${work_dir}" -D "${install_dir}" \
  -L "${iso_label}" -o "${out_dir}" iso "${iso_name}-${iso_version}-dual.iso"
```

これにより，CD のラベルに negix Live/Rescue CD と設定されるようになりました．

[*5] このディレクトリを HTTP 経由にて公開することで，そのまま公開リポジトリとして利用することもできる．

パッケージリストの修正

次に，LiveDVD にインストールされるパッケージの一覧 packages.both に，mikutter LiveDVD に必要なパッケージ名を追記します．mikutter LiveDVD に必要なパッケージの一覧は以下のようになります．

packages.both への追記

```
xorg
lxdm
xcompmgr
fluxbox
fcitx-mozc
fcitx-configtool
fcitx-gtk2
fcitx-gtk3
fcitx-qt4
ttf-vlgothic
alsaplayer
alsa-plugins
alsa-utils
alsa-firmware
tmux
sakura
firefox
evince
feh
mikutter
```

これらのパッケージが，それぞれのアーキテクチャ用イメージ内にインストールされます．もし，特定のアーキテクチャのパッケージにのみ追加したい場合は，それぞれのアーキテクチャに対応した packages ファイルに追記してください．

Live 環境内ファイルの準備

最後に，ベース環境に上書きするファイルやディレクトリを root-image 追加していきます．アプリケーションの設定ファイルの編集や各種ファイルの追加はここでおこないます．mikutter LiveDVD にて編集・追加しているファイルの一覧は表 2の通りです．

表 2　Live 環境に上書きするファイル一覧

ファイル/ディレクトリ名	編集内容
root-image/etc/hostname	デフォルトのホスト名を negix に変更
root-image/etc/locale.conf	デフォルトのロケールを ja_JP.UTF-8 に変更
root-image/etc/lxdm	lxdm の設定 (自動ログイン，ログインスクリーンの変更)
root-image/etc/skel/.fluxbox	fluxbox の設定
root-image/etc/skel/.config	fcitx，sakura の設定
root-image/opt/mikutter	うすい本，壁紙等
syslinux/splash.png	ブートローダの背景画像の変更

customize_root_image.sh の編集

ベース環境の準備が完了した後にベース環境内で実行されるスクリプト customize_root_image.sh の編集も合わせておこないます．このスクリプトでは，ユーザアカウントの追加やタイムゾーンの設定，systemd によるサービスの起動に関する設定をおこなうことができます．mikutter LiveDVD では，以下の項目について修正してあります．

- デフォルトロケールを ja_JP.UTF-8 に
- タイムゾーンを JST に
- ユーザアカウント miku を追加
- systemd のデフォルトターゲットの変更 (multi-user.target から graphical.target)
- dhcpcd, lxdm の有効化

root-image/root/customize_root_image.sh の編集

```
sed -i 's/#\(ja_JP\.UTF-8\)/\1/' /etc/locale.gen
locale-gen
ln -sf /usr/share/zoneinfo/Japan /etc/localtime
usermod -s /bin/zsh root
cp -aT /etc/skel/ /root/
useradd -m -p "" -g users \
  -G "adm,audio,floppy,log,network,rfkill,scanner,storage,optical,power,wheel" \
  -s /bin/zsh -u 3939 miku
chmod 750 /etc/sudoers.d
chmod 440 /etc/sudoers.d/g_wheel
sed -i "s/#Server/Server/g" /etc/pacman.d/mirrorlist
systemctl enable graphical.target pacman-init.service dhcpcd.service lxdm.service
```

ビルド

全てが完了したら，mikutter LiveDVD をビルドします．

```
master@hachune# ./build.sh -N negix -L NEGIX_201306
```

もし ISO イメージが生成されない等の問題が発生した場合は，-v オプションを付けて原因を探ってみてください．

おわりに

今回は，Arch Linux の LiveCD を元に mikutter LiveDVD を作成する方法を解説しました．この mikutter LiveDVD の構築とだいたい同じ手順にて，オリジナル RescueUSB メモリの作成等もおこなうことができます．また，この LiveDVD 作成に関する情報を元に作成したのが，オープンソースカンファレンス 2013 Tokyo/Spring の mikutter ユーザ会ブースにて展示した，有線 LAN ポートに接続し PC を起動するだけで mikutter 環境が利用できる「OpenBlocks AX3 を用いた mikutter 環境展開システム」となっています．

これからも mikutter と mikutter ユーザ会をよろしくお願い致しますヾ (@⌒ー⌒@) ノ

Plugin の新機能

@toshi_a

昨今の mikutter のプラグインに依るエコシステムの進化はご存知のとおりです。今回は最近導入されたプラグイン周りの機能を紹介していこうと思います。

1. プラグイン定義ファイル

0.2.1 から、プラグイン定義ファイルが導入されました。これからはすべてのプラグインに定義ファイルを同梱することが 推奨 されます。定義ファイルがあることによって、以下のようなメリットがあります。

- 想定する **mikutter** のバージョンを指定できる。 今のところ、mikutter のバージョンが古い場合にプラグインを起動できないようにすることができる。
- プラグインの依存関係を明示できる。 いままで、プラグインは各々がイベントを使ってゆるい連携しかできませんでしたが、あるプラグインがないと動作しないといった指定が可能になりました。

  他にも、名前や作者名などのメタ情報を入れることができます。0.2.1 では単に存在するだけでしたが、0.2.2 では条件が満たされていなければ実際にロードされないようになりました。実際の書き方等は、vol.1 に寄稿した「Writing mikutter plugin」に追記したので、参照してください。

> Writing mikutter plugin http://toshia.github.io/writing-mikutter-plugin/

2. 新しい Plugin クラス

0.2.2 では、三年ほど前に書いた Plugin クラスがほとんど書きなおされ、以下の 4 つのクラスに分かれました。

- **Plugin** プラグインの実体
- **Event** イベントの種類
- **EventListener** on なんとか
- **EventFilter** filter なんとか

Plugin 自体のインターフェイスにはほとんど追加しかないので、この変更自体気づいた人は少ないでしょう。

2.1. イベントのインスタンス

イベントは種類ごとに Event のインスタンスを持つようになりました。例えば mentions イベントのインスタ

ンスは以下のようなコードで取得できます。

```
e = Event[:mentions]
```

Event#call を呼べばイベントが発火されますし、 Event#filtering は引数をフィルタリングして返します。これらのことは Plugin.call や Plugin.filtering でも今までどおり実行できます。

2.2. イベントの優先順位

興味深い新機能は Event#priority です。イベントの優先順位という概念が追加されました。今まで mikutter は、発生した順番にイベントを処理していましたが、突発的に大量のイベントがキューイングされ、あとまわしでもいいイベントの処理によって、ユーザに依る入力イベントがなかなか処理されないことがありました。

priority は、イベントの種類ごとの優先順位です。具体的に言えば、 Event#call 時に Delayer の優先順位としてこａの値が渡されます。以下は開発版のコンソールでの出力です。

```
>>> Event[:mentions].priority
:routine_passive
>>> Event[:favorite].priority
:ui_favorited
```

mentions のような一般的なイベントには、標準の :routine_passive が割り当てられていますが、favorite のような突発的に大量のイベント発生が見込まれるイベントは、 :ui_favorited という最も低いプライオリティが割り当てられています。このことによって、ふぁぼ爆撃を受けている間でも、リプライは遅れなく受け取ることができるのです。

これは最適化の一環で試験的に導入された機能です。他にも、イベントのプロトタイプ宣言なども試験的に追加されています。

2.3. イベントのデタッチ

イベントリスナを定義する on_*メソッドの戻り値が、EventListener のインスタンスになりました。プラグインから使えるメソッドは detach くらいですが。Detach メソッドはそのイベントリスナを削除するメソッドです。

```
Plugin.create :test do
  e = on_update do |s, ms|
    e.detach
    # 処理...
```

```
  end
end
```

filter_*メソッドも EventFilter を返すようになりましたが、こちらにも同じ意味合いの detach メソッドが定義されています。

3. プラグイン DSL 拡張

3.1. メソッドを追加

以前から、 Plugin#settings のような、あるプラグインのためのメソッドを Plugin が提供してしまっているというものはありました。サードパーティからこのようなメソッドを追加するにはどうするべきでしょうか。

いままでは、Plugin にモンキーパッチしてメソッドを増やすという事をやっているプラグインもありましたが、そもそもロード順が不定なため、そのメソッドを使った他のプラグインが先にロードされると起動できませんでした。0.2.1 からは spec ファイルが適切に設定されていれば、依存しているプラグインが先にロードされるのでこのような問題はなくなっています。

さらに 0.2.2 では、モンキーパッチを使わずに DSL を拡張するための仕組みが追加されました。それが defdsl です。実際に、Activity プラグインでこれが使われています。

```
defdsl :activity do |kind, title, args = {}|
  Plugin.call(:modify_activity,
              { plugin: self,
                kind: kind,
                title: title,
                date: Time.new,
                description: title }.merge(args)) end

defdsl :defactivity do |kind, name|
  filter_activity_kind do |data|
    data[kind] = name
    [data] end end
```

メソッド定義に似ていますし、中でやっていることもほとんどモンキーパッチと変わりませんが、いちいちプラグインクラスを開き直さなくても定義できるのがメリットです。例えば modify_activity イベントは以下のようなコードで発生させることができますね。

```
activity :system, "お茶湧いたよ"
```

3.2. スコープ

defdsl のもうひとつのメリットはスコープにあります。 先程例示したコードの 3 行目、defdsl 内の self とは何でしょうか。実はこれは、activity メソッドを呼んだプラグイン自身のスコープになります。だから、defactivity ではフィルタを新しく登録していますが、これは Activity プラグインではなく、呼び出し側のプラグインに登録されます。当然、メソッド呼び出しなんかもそちらのプラグインのものが呼ばれるので注意しましょう。

これだけならモンキーパッチと同じですが、defdsl は外のローカル変数を参照することができます。クロージャだから当然ですが、これは重要なテクニックです。

```
x = 1

defdsl :setx do |num|
  x = num
end

defdsl :plusx do |num|
  num + x
end

plusx 1 # => 2
setx 2
plusx 1 # => 3
```

3.3. いつ使うのか

標準プラグインも移行しきっていないのですが、Plugin にメソッドを追加したいすべてのケースでこれを活用すべきです。 具体的には、以下のような場合でしょうか。

- activity メソッドのような、他のプラグインからイベントを発生させて受け取るという通信をする場合に、そのイベントの呼び出しを隠蔽または簡略化したい場合
- defactivity メソッドのような、他のプラグインのイベントリスナやフィルタの定義の面倒を見てやりたい場合

3.4. デメリット

DSL メソッドは便利ですが、良いことばかりではありません。

- プラグイン同士の依存が強くなりすぎる イベントを発生させるだけなら、受取り側がいなくても無視されるだけで問題はありませんから、依存関係にする必要はありません。しかし、DSL 拡張を使うという事は、明確に依存することになってしまいます。
- イベントを伴わなずに通信できてしまう 本質は単純なメソッド呼び出しなので、呼び出し元のプラグインの時間を使って処理を実行してしまいます。このことはデバッグを困難にし、他のプラグインがフィルタやフックを用いてデータを利用できなくなることを意味します。

以上のような理由から、基本的には DSL 拡張は使うべきではありません。一方で、setting, tab, timeline, activity, defachievement 等、DSL 拡張をうまく使った強力な機能が既にたくさんあります。うまく使えば、強力なメタプラグインを作成することができるでしょう。

4. まとめ

今回は、プラグイン関連の新機能についていくつか解説しました。プラグイン機能は最も重要な機能で、プラグイン同士の連携こそが mikutter の強みです。mikutter の最終目的は、開発者が自由になんでもできるエコシステムを構築することです。かつてそのことが Twitter を成長させたように、mikutter はこれからも成長していくことでしょう。

あとがき --brsywe

さて今回で 4 巻目となる『Too late.』お楽しみ頂けただろうか。今回も幾人もからありがたい支援を受けつつ、原稿があやうくておくれかけつつ発行に至ることができた。年末から年始にかけていろいろと忙しくなるので次号がはたして冬に出るかどうかは不明である。次に会う時まで。

校正者　@ch_print のあとがき

あれ、なにも書いてなくね。

イラスト

@soramame_bscl 表紙　@toshi_a 31 頁

@shijin_cmpb 裏表紙（右）

缶バッジ用イラスト

@soramame_bscl , @iori4213

マグカップ用イラスト

@soramame_bscl , @shijin_cmpb

——次号予告と寄稿者募集—　　—

原稿提出期限：11 月 20 日　但し編集者都合により発行を次の夏に遅らせる可能性あり

発行予定：コミックマーケット 85 または こみっくトレジャー23

問合せ先：@brsywe

奥付

発行日：冊子版：2013 年 8 月 3 日(OSC Kyoto@Kansai) – 8 月 12 日(コミックマーケット 84)

PDF 版：2013 年 8 月 18 日

発行：mikutter の薄い本制作委員会

発行者： @brsywe 西端の放送局内喫茶室長

連絡先： brsywe　[at]　hotmail.co.jp

ご意見・ご感想はお気軽にどうぞ。

mikutter の薄い本制作委員会ウェブページ

http://home1.tigers-net.com/brsywe/mikutter.html

mikutter の薄い本制作委員会では、Amazonギフト券・DMMギフト券による金銭面の支援を受け付けております。

もし、あなたがこの薄い本を読んで、何かしら満足感を得られたなら、送って貰えるとその満足感を誰かと共有できるかも。

上のウェブページを御参照ください。