mikutterの薄い本 vol.5

としあくんちのておくれ事情

The too late situation of toshia's family

ておくれなとしぁと一緒にTLにいる未来を賭けて、ておくれたちの
「mikutterの薄い本争奪戦」が、いま、始まる――！！

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」にすべて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。なお先の URL は 2014 年 4 月以降変更される可能性があります。その場合には閲覧者が許諾条件を探して従う必要があります。

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　当該書籍はコミックマーケット 85 において 200 円で頒布されたもので、次回の OSC 京都に於いても有償で頒布される予定のものです。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

執筆日　平成 25 年 12 月 12 日　「戦略的撤退を選択しつつ」　@ch_print（印刷室長）

前書と反省とお知らせ

・執筆者・編集者

今号では @brsywe が修論に追われ死にかけているため前書と反省さえも わたし @ch_print に投げられてしまった。印刷室長とはいったい誰なのか。その真実を知る人は喫茶室長とわたししかおるまい。

1. 喫茶室長現況

今号においてはついに紙面にさえ現れていない喫茶室長。どうしたことなのか詰問したところ、以下の回答が得られた。

「修士課程の修了期限を間近に控え、就活と美少女ゲームに時間を費やしすぎ、そのツケを払っている最中である。読者各位の期待を裏切る結果となったことを深くお詫びし、今後の再発防止に努めたい。」

なお聞くところによれば就職先がようやく決まったそうである。来夏以降この薄い本はどうやって刊行されるのか。今後の課題の一つである。

2. 頒布関連

前号のときには初めてコミケに当選したということもあってかそうそうに売り切れてしまった。余裕をもった数を作って搬入したにもかかわらず、である。自分のサークルが当選したときには委託するときよりも少し多めにしたほうがいいのかもしれない。

3. ウェブページ

Shift-JIS 叩きの影響を受けたのが mikutter の薄い本制作委員会のウェブページである。知識が不十分な喫茶室長が Shift-JIS でウェブページを作っている。先の就職の影響もあって現在契約しているプロバイダを替えることが決まっている。レンタル鯖をどうするかも今後の課題の一つである。そのため、今 PDF 版を公開したり、薄い本の頒布告知をしたりしているウェブページは移転することになる。移転後の URL はわたし @ch_print が告知することとなるのでﾌｫﾛｰするなりリストに入れておくなりして確認していただければ幸いである。

4. 今後の予定

現在検討しているのは mikutter の薄い本 vol.6(C86)と、mikutter の薄い本別冊創刊号である。後者について、mikutter に関連しない Twitter 全般の諸問題やウェブサービスについての薄い本としたいと考えている。すでに若干名に記事の執筆依頼をしているがさてはてどうなることやら。

5. さて本題

今号 vol.5『としぁくんちのておくれ事情』。タイトルは見てのとおりである。死にかけの喫茶室長がやってた美少女ゲームのタイトルのもじり。

＜目次＞　著者・表題・ページ

6. ＜付録＞缶バッジ図案

上 @shijin_cmpb

下 @soramame_bscl

7. それでは

　かくいうわたしも忙しいので駄文を連ねる暇すらない。次ページから生粋のておくれによるておくれな記事である。楽しんで行っておくれ。

本誌のイラスト等提供者

題字 @T_H_sister_

表紙 @soramame_bscl

後書,裏表紙 @shijin_cmpb

8. PDF 版追記

なお、喫茶室長は修論ておくれたらしい。

なれる！ておくれ

ぺんぎんさん（@penguin2716）

1　はじめに

読者の皆様は「ておくれ」とは何かご存知でしょうか．もしあなたが「ておくれ」と「手遅れ」の違いがわからないようであれば，この記事を読まずにページをめくりましょう．さもないと，あなたを真の ておくれ になってしまうかもしれません…．自分が ておくれ になってもかまわないという方は，ておくれ の素質がありますのでそのまま読み進めてください．なお，この記事で論じている「ておくれ」は著者視点で認識したものですので，他の ておくれ との認識は異なる可能性があります．ご了承ください．

2　ておくれ になりたい！

「だめだ…ここは ておくれ すぎた…」 – @toshi_a

2.1　ておくれ って何

Twitter で「ておくれ って何ですか」と聞かれることがよくありますが，安全のため，Twitter では具体的なことを言わないように気をつけています．「ておくれ とは何か」を知ってしまうと，もうあとには戻れないからです．ただ電車に乗り遅れたり，ただ寝坊したりするだけなら誰にでもできますが，彼らはそれをごく自然にやってのけます．いかに巧妙に ておくれ るフラグを立て，いかにエレガントにそのフラグを回収するか，そこに ておくれ るという難しさがあります．

2.2　ておくれ になるには

ておくれ になるために，まずは ておくれ の元帥とも呼ばれるとしぁさん（@toshi_a）をフォローしましょう．次に，としぁさんが追加されているリストを表示して「ておくれ」いう文字列で検索します．そのリストに追加されている人をフォローして「俺は ておくれ になるという固い決意をした」とつぶやけば準備完了です．既にあなたから ておくれ のオーラが漂っていることでしょう．

図 1　mikutter を開発しているのに OSX で mikutter を動かせない人の例

2.3　mikutter のインストールは必須？

mikutter のインストールは必須ではありません．実際，mikutter を利用していない ておくれ もたくさんいます．図 1 に示すように，mikutter を開発しているのに OSX で mikutter を動かせない人とかも中にはいますのでご安心ください．ここまで読んで「俺は mikutter を使っていないから ておくれ じゃない」と思ってしまった人には残念なお知らせですが，mikutter を使っていないからと言って，あなたが ておくれ じゃないとは証明できません．諦めましょう．

3　ておくれ る練習をしよう

「あ、ておくれ だから俺のことではないのか」
– @toshi_a

3.1　レベル 0：「俺は ておくれ じゃない」編

さて，多くの ておくれ たちは「俺は ておくれ じゃない」と Twitter でつぶやきます（ツイート時には「ておくれ」を「手遅れ」と誤変換しないように気をつけてください）．これは，「俺は ておくれ じゃ

ない」と自分をアピールしているのですが，本当に おくれ じゃない人は「俺は おくれ じゃない」などというツイートはしないので，彼らは立派な おくれ です．この際，自分のことを「おくれ」だと感じているか「普通の人」だと感じているかは問題ではありません．「おくれ」という認識がそこに介入している時点で，そいつは おくれ なのです．

3.2 レベル 1：「乗った電車が反対に向かったんだけど」編

さらなる おくれ に向けて，うっかり反対側の電車に乗ってみましょう．「うっかり」乗ってしまうというところがポイントです．わざと反対側の電車に乗ってしまった場合は，申し訳ない気持ちで「終電間に合った」とツイートした後に「あれ，この電車…反対側…だと…？」とツイートしましょう．おくれ 力を人為的に上げることができます．ぷよぷよの連鎖みたいなものです．

3.3 レベル 2：「えっ UTC じゃなかったの」編

普段からパソコンを利用している皆さんのタイムゾーンはもちろん UTC だと思いますが，レポートの提出なんかは JST で指定されることがあります．この際，「えっレポートの提出 UTC じゃなかったの wwwww」のようにつぶやくと「あっこいつ おくれ だ」と思われることができます．他人に おくれ だと思われてこその おくれ です．自分のことを おくれ だと認識されなければ，その人がいくら おくれ でも，おくれ にはなれないのです．提出時間にギリギリ間に合わなかった場合は，「メールのタイムスタンプを見ると遅れてる？利用してる NTP サーバがずれてるんじゃないですか？？」と言いましょう．提出時間に大幅に間に合わなかった場合は，仕方がないので「提出時間が俺を置き去りにしたのでちょっと道に迷ってたらもうこんな時間かよ」とだけつぶやいて枕を濡らしましょう．

3.4 レベル 3：「気がついたらテスト間に合わなかったよ」編

あなたが学生なら，テスト前日に夜遅くまでテスト勉強をしていて（だ，断じて，断じて深夜アニメを見ていたわけではないんですよ！！！）起きたらテストに間に合わなかったという経験があると思います．これではただの おくれ ですが，効果的に おくれ たことを皆様に伝えるため，「俺の起床時間が遅くてテストに間に合わなかったんじゃない，遅かったのは深夜アニメの放送時間だ」というようにツイートしましょう．

- 寝坊した
- テストに間に合わなかった
- アニメを見ていた
- 何言ってんだこいつ

という 4 つの要件がフォロワーさんに適切に伝わることでしょう．

4 「エレガントに おくれ る」ということ

そうです，ただ おくれ るだけでは，あなたはただの おくれ 止まりです．もっとエレガントに おくれ ることで，さらなる おくれ の高みに辿り着けるのです．エレガントに おくれ るためには，最初のうちは少し工夫が必要です．「どうしたらエレガントに おくれ ることができますか？おしえてください。」（http://theinterviews.jp/toshi_a/1351497）にもあるように，言い方も非常に重要ですね．

> *「おくれ るって、かっこいいことじゃないんじゃないでしょうか。何をしようとも、もう起こってしまったことは おくれ です。つまり、いかにエレガントに語るかが重要だと思います。」*
>
> *– @toshi_a*

加えて，そのツイートで「こいつ おくれ たんだな」と思わせることも重要です．上述したように，この世界は他人から認識された姿が評価対象とされますので，相手に分からないように おくれ ても仕方がありません．それでは，おくれ に付加価値をつけるような，皆さんのクリエイティビティあふれる おくれ を期待しています．

mikutter の楽しみ方

@tsutsuii

1. はじめに

作者の@toshi_a さんにより mikutter 開発が公式に始まったとされる 2009 年 12 月 25 日からちょうど 4 年。そして、2012 年 1 月に vol.1 が発行された mikutter の薄い本も今回で 5 冊目となりました。作者の思惑とは直接関係なくユーザーの呼びかけにより薄い本が 5 冊も発行され、なおかつそれがコミケで頒布そして完売してしまうようなクライアントは mikutter 以外にはありません。これは、vol.4 に寄稿した「mikutter はなぜ人をおくらせるのか」の記事で紹介したとおり、mikutter の持つ魅力が様々な層のユーザーを惹きつけているからと言ってよいでしょう。

かく言う私も mikutter の薄い本 vol.2, 3, 4 と記事を書かせてもらって~~そおくれ認定されて~~いますが、改めて過去の薄い本の記事を見直してみると、さすがに記事執筆に名乗りを挙げる方々だけのことはあり選りすぐりの~~そおくれな~~濃い内容の記事が半数以上を占めています。そもそも Twitter クライアントの薄い本などというものを手にする方はそれだけで十分に~~そおくれの~~素質があるわけで、書く側も存分におくれた記事を提供できれば互いに Win-Win であるとは言えます。

とはいえ、「mikutter ってなんなんだろう」という思いで薄い本を読む方もいれば、コミケでふらりとブースに立ち寄って本を手にする方も少なからずいるのではないかと思います。また、mikutter を使ったことのない方、そしてそもそも Twitter クライアントというものを使ったことがない方を対象に、mikutter の導入から使い方、そしてより深い部分へ~~そおくれていく~~楽しみを広げていくまでを紹介することも、薄い本の記事としての意義はあるのではないかと思います。

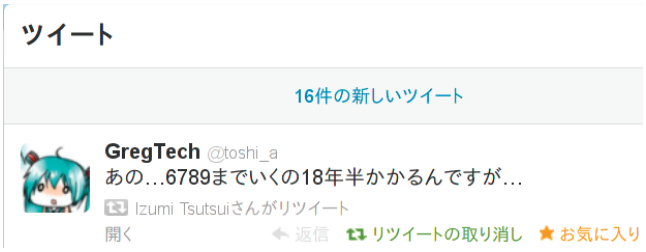
というわけで、今回は前回までのおくれ一色な記事とはちょっと趣向を変えて、mikutter そのものの紹介と、過去の薄い本の記事をより深く理解して~~そおくれる~~楽しむための導入部分としての説明を書いてみようと思います。

2. mikutter 導入の前に

「Twitter を始めてみよう」という場合、たいていは web ブラウザで http://twitter.com/ の Twitter 公式 URL にアクセスしてユーザー登録し、そのまま公式 web 上で Twitter を使うことになると思います。一方で、世間には mikutter を始めとしてサードパーティーによる Twitter アクセス専用のクライアントが多数リリースされています。

これは、過去の公式 web インターフェースがかなり使いづらかった（リプライがどのツイートに対する返信かもわからなかった）というのもありますが、現状の公式 web あるいは公式クライアントでは Twitter を楽しむ上で足りない部分あるいは使いづらい部分があり、それを解決したいという思いが各専用クライアントリリースの動機付けになっています。

公式 web と専用クライアントとの大きな違いとしてはツイート表示のリアルタイム性が挙げられます。公式 web では一定時間ごとに「xx 件の新しいツイート」という表示のみがされ、そこをクリックすることで新規ツイートが表示されるようになっています。



公式 web の新規ツイート通知

一方、mikutter を含む多くの専用クライアントではタイムライン上のすべてのツイートを次々とリアルタイムで表示する形式になっています。

Twitter を一方的な情報収集ツールとして使うのであれば公式 web でも特に不都合はありませんが、親しいフォロー・フォロワーが増え、友人のツイートに対するツッコミやお気に入り（ふぁぼ）を飛ばしたり自分宛のメンションに対して反応を返したりという会話的な使い方が増えると、専用クライアントによるリアルタイム表示のほうがテンポのよい会話を行うことができます。その際、リアルタイムのタイムライン表示ではツイート数が多くなるとツイートがどんどん流れていってしまうことになりますが、専用クライアントではそれに対応した通知方法や返信入力方法などのインターフェースがそれぞれ工夫されており、自分にあったツールとしてのクライアントを選ぶことができます。

また、公式 web では無料サービスの宿命として広告ツイートが挿入されるほか、往々にしてインターフェースが強制的に変更される上に、カスタマイズもできないという問題があります。この点についても、自分の感性

にあったクライアントを選べば気に入った UI を使い続けることができますし、たいていのクライアントは表示やキーバインドなど多くのカスタマイズ機能を備えています。

さらに、Twitter クライアント特有の利点(?)として、各クライアントの開発者は多くの場合 Twitter のヘビーユーザーでもあるため、開発者をフォローすればそれぞれのクライアントのポリシーを直接見聞きすることもできますし、会話の機会があればユーザーの感想を直接伝えることができる、ということも挙げられます。

3. mikutter のインストール

次に、専用クライアントとしての mikutter の特徴を説明する流れですが、そこは mikutter の薄い本の記事ですのでまずはとっとと mikutter をインストールしてみて使いながら見ていくことにしましょう。

ただし、mikutter がターゲットとしているのはいわゆる「Linux とか」と呼ばれる OS(詳細については薄い本 vol.2 の「mikutter と NetBSD のておくれな関係」の記事を参照)であり、Windows や Mac OS X、スマフォやタブレット端末では簡単に試すことはできません。mikutter のために Linux 系 OS をインストールする、あるいは仮想マシンを用意するというておくれな高度な選択肢もありますが、とりあえず専用クライアント自体使ったことがないという方は mikutter の README にも紹介されている夜フクロウ、Janetter、Tweetbot、ShootingStar といったクライアントを試してみるとよいかと思います。また、お試しとして後述する mikutter wiki で紹介されている LiveDVD を使うという手もあります。

mikutter のインストールに当たっては、まず mikutter の FAQ ページ http://mikutter.hachune.net/faq の以下の「動作条件」にある文章に注意しましょう。

> **人柱気質　少々**
> **ユーモア　このページを読んで気分を害さない程度**

Linux 系の OS を使っているユーザーであればオープンソースで公開されているソフトの対応に文句をつけるような方はそういないとは思いますが、mikutter は作者の@toshi_a さんほぼ一人で開発されており、インストールやサポートよりも mikutter 本体の開発が優先されています。また、mikutter 内部には各所に上記 FAQ ページと同様なユーモア(?)を含んだ機能が散りばめられています。この辺の感覚が自分に合わないと感じた方は残念ながらこのあたりで引き返したほうが無難かもしれません。

実際のインストールについては mikutter の download のページ http://mikutter.hachune.net/download のページにまず目を通しましょう。download ページの「パッケージマネージャを利用する」にあるように、現在では Linux のメジャーなディストリビューションおよび他の「Linux とか」系の OS でも mikutter が公式非公式含めパッケージシステムとして提供されていますので、まずはそれを使用しましょう。

各 OS での詳しいインストール方法については download ページからリンクされている mikutter wiki の http://yuzuki.hachune.net/wiki/MikutterInstallBattle 「mikutter のインストール」のページにありますのでそちらを参照してください。具体的には、Ubuntu や Linux Mint であれば apt-get install mikutter といったコマンド、NetBSD などでは pkg_add ruby193-mikutter といったコマンドを端末ウインドウ上で実行することになります。

mikutter wiki の InstallBattle のページ

4. mikutter の起動

mikutter がインストールできたら次は早速起動です。Ubuntu や Linux Mint、NetBSD などのパッケージマネージャを使ってインストールをした場合はたいてい mikutter 起動用のスクリプトもインストールされるため、mikutter の起動はコマンドライン上で“mikutter”のコマンドをタイプするだけです。

mikutter を起動すると、まず Twitter 認証用の「mikutter ログイン」のウインドウが出てくるはずです。

mikutter ログインの認証ウインドウ

ログインウインドウの表示にしたがってリンクをクリックするとデフォルトのブラウザで Twitter 認証ページが開

きます。そこで mikutter で使用するユーザー名（スクリーンネーム）とパスワードを入力すると 7 桁の暗証番号（PIN）が表示されるので、その数字をログインウインドウの「暗証番号」欄に入力して OK を押します。これで認証は完了で、mikutter のウインドウが表示されるはずです。認証結果はホームディレクトリの .mikutter ディレクトリ内に保持されるので、次回以降の起動では認証は不要です。

ログインウインドウが表示されない場合は、まずはマシンの時計の設定を確認してください。時計が実際の時間から大きくずれていると認証に失敗することがあります。その他のトラブルについては各 OS ディストリビューションに詳しい人に助けを請いましょう。

mikutter の初回起動時の表示（タイムライン）は以下のような画面です。

mikutter 初期起動ウインドウ

一番上にツイート入力のウインドウが、そして右側に各種タブが縦に並んだ状態になっているかと思います。公式 web である程度 Twitter を使っている方なら感覚的にわかると思いますが、「t」のタブがフォローしている人たちのツイートおよび自分宛のツイートが表示されるメインのタイムライン、矢印のタブが自分宛のメンション表示、「d」のタブはダイレクトメッセージ、以下アクティビティ、検索、フォロー、フォロワー、保存した検索等々の各種タブが並びます。

「t」のタイムラインタブを選択すると、まず mikutter のマスコットでもある @mikutter_bot さんが各種機能の助言をしてくれていると思います。これは「実績」と呼ばれるお知らせ機能で、助言された機能を実行すると晴れて実績解除となり @mikutter_bot さんから祝福のメッセージをいただけます。なお、この状態ですでにタイムラインのリアルタイム表示は有効になっており、受信されたツイートが次々と上から下へ流れていくはずです。

また、自分のツイートがふぁぼられた場合や RT された場合はツイートがタイムラインの先頭に持ち上げられ、ふぁぼられ数と RT 数を表示してくれます。

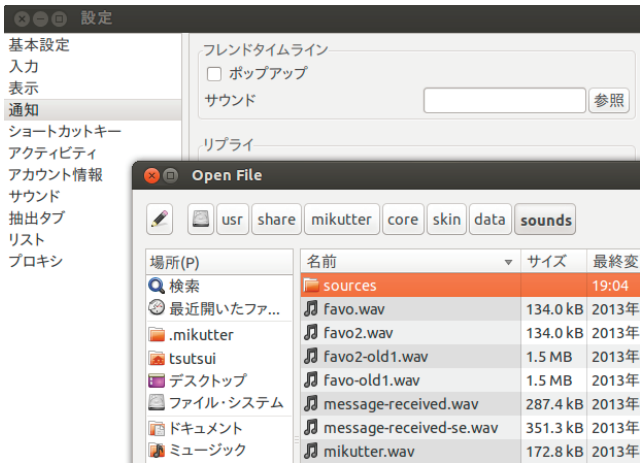
ツイートのふぁぼられ＆RT られ表示

5. mikutter の設定

とりあえず初期起動状態の mikutter を画面のすみに配置してタイムラインをぼーっとながめる、でもよいのですが、前述のようにツイートのリアルタイム更新表示を使う場合には通知機能の設定がほぼ必須です。まずは上記のタイムライン画像で @mikutter_bot さんが助言してくれているとおりに画面右下にあるレンチのアイコンをクリックすると「設定」のウインドウが開きます。専用クライアントならではの各種設定項目が多数あるため一つ一つの説明は省略しますが、まずはウインドウ左側の「通知」を選びましょう。そこでツイート受信やリプライ受信時など各種イベント時の通知方法を設定できます。

「ポップアップ」にチェックをすれば（使用しているディストリビューションのデスクトップ環境がサポートしていれば）画面上にポップアップウインドウで通知が表示されます。「サウンド」の「参照」ボタンを押すと、イベント時に再生する音声ファイルを指定できます。デフォルトでは音声ファイルは何も指定されていませんが mikutter の配布中にはデフォルトでいくつかのサンプル通知音声が入っていますのでそれを指定しましょう。ディストリビューションにもよりますが、音声ファイルは /usr/share/mikutter/core/skin/data/sounds あるいは /usr/pkg/share/mikutter/core/skin/data/sounds

といったディレクトリにインストールされているはずです。



通知用サウンドファイル指定画面

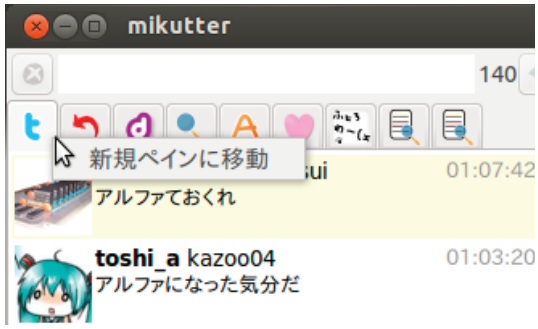
「フレンドタイムライン」のサウンドとして「mikutter.wav」を指定すれば、ツイート受信のたびに「みくった～♪」の音声、「ふぁぼられたとき」のサウンドとして「favo.wav」を指定すれば、ツイートをふぁぼられるたびに「ふぁぼっ♪」という音声で mikutter さんが通知をしてくれます。それなりの数のツイートが流れるタイムラインであれば、この通知設定だけでもタイムラインがだいぶ楽しいものに変化したことが実感できると思います。

「通知」以外ではとりあえず「表示」のところでフォントやタブの位置を変更するくらいで、あとはおいおい慣れてから内容を確認してみればよいと思います。

「設定」ウインドウ以外で行うカスタマイズとしてはマルチペイン表示設定があります。前述のようにタイムラインの他にリプライ、検索、リストなど多数のタブがありますが、デフォルトの 1 ペインで必要に応じてタブを選択するもよし、サブ液晶に mikutter を全画面表示して複数のタブをマルチペインで複数表示するもよし、各自の Twitter スタイルに応じて画面を構成できます。

マルチペイン表示をするには各タブを右クリックして表示されるメニューの「新規ペインへ移動」を選択します また、各タブをクリックしてドラッグすることでタブ同士、あるいはペイン間でのタブの移動をすることもできます。

マルチペインではタイムライン、リスト、検索を同時に表示するような使い方になるかと思いますが、特にリストについては mikutter 特有の機能として「リスト上のツイートもタイムライン同様にリアルタイムで表示される」という機能があり重宝します。タイムラインのツイートが多くなりすぎた場合に一定の人たちをリストに退避する、あるいは逆にメインのタイムラインで流れてしまっても見逃したくない人をリストに入れておく、というのがリストの一般的な使用法ですが、mikutter のマルチペイン表示ならばリストのツイートに対してもリアルタイムで即反応することが可能です。



タブ右クリックによる新規ペインメニュー

6. 追加プラグイン導入

ここまで mikutter インストール後の設定を書いてみて、こういった説明が公式に存在しないことに改めて驚いていたりしますが、とにかくここまでくれば一通りの Twitter 生活を送ることはできるかと思います。

が、mikutter でできる設定はこれだけではありません。そうです、mikutter の最大の特徴でもあるプラグインがまだ残っています。とは言っても、mikutter を使う上でのお勧めプラグインについてはすでに薄い本 vol.4 に @penguin2716 さんが書かれた記事「今すぐインストールすべき mikutter プラグイン 10 選」で詳しく説明されていますのでそちらを参照してください。とりあえずは「ておくれ初級」で紹介されている Display Requirements プラグイン、クライアント名表示プラグイン、アカウント名補完プラグインあたりが定番です。

これ以外にも mikutter 用プラグインは日々作成されています。前述の mikutter wiki のページのプラグイン項目を定期的にチェックすると予想もつかないようなプラグインで一層ておくれた mikutter ライフを送れるようになるかもしれません。ただし、前述の記事を書いた @penguin2716 さんのように調子に乗ってプラグインを 50 個以上もインストールしたりすると mikutter の起動が相当重くなるだけでなく何もかもておくれてしまいますので、何事もやりすぎには注意しましょう。

ぺんぎんさんのておくれプラグインツイート

7. としぁさんとておくれな愉快な仲間たち

mikutter の設定とプラグインの導入まで終われば、mikutter による~~ておくれ~~ Twitter 生活としては一通りの用意が整ったといえますが、mikutter 作者の@toshi_a さんを始めとした mikutter 界隈のユーザーをフォローしてみると、より mikutter をとりまく~~ておくれ~~ユーザーの間で行われている mikutter 流(?)の Twitter の楽しみ方を感じることができるかもしれません。

@toshi_a さんといえば、なんと言っても有名なのがその「ふぁぼられ数」です。今から 1 年ほど前の 2012 年 10 月にはお気に入り数集計サイトである Favstar.fm の集計で見事 100 万ふぁぼられを達成していました。この、@toshi_a さんと mikutter のまわりのふぁぼ・ふぁぼられについては、薄い本 vol.4 の記事でも紹介した@toshi_a さん自身による「ておくれ、ふぁぼ、mikutter」の記事 http://dl.dropboxusercontent.com/u/7993463/f2t-advent.html に詳しく解説されていますのでそちらを参照してもらうとして、実際に@toshi_a さんがふぁぼ爆撃を受けている様子は@toshi_a さん自身の反応を含めて~~おくれて~~おり新たな文化を垣間見ることができます。

大変残念なことに、@toshi_a さんはあまりにも大規模にふぁぼられ過ぎてしまったせいか前述の Favstar.fm の有料会員であったのにもかかわらずアカウントを BAN されるという憂き目に遭い、現在では正確なふぁぼられ数を把握することはできなくなっています。しかし、Favstar.fm に代わる新たなふぁぼられ観測サイトとして aclog (http://aclog.koba789.com/) というサイトが立ち上げられています。このサイトでの@toshi_a さんのふぁぼられ状況 http://aclog.koba789.com/toshi_a/timeline を横目で見つつ mikutter タイムラインを流せば、より効果的なておくれ観測ができるかもしれません。

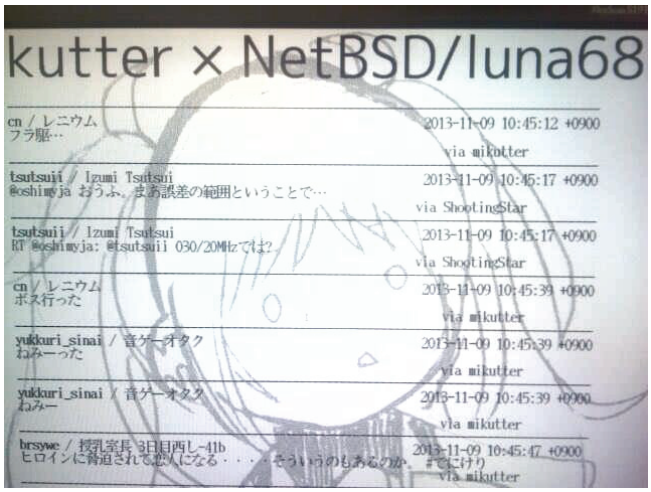
aclog による としぁさんの最近のふぁぼられ表示例

8. エクストリーム mikutter

mikutter を使った生活を続けていると次第にあらゆる状況で mikutter を使いたくなるのか、はたまたユーザー自身がておくれていくのか、あらゆる場所やあらゆる機会で mikutter を動かすというエクストリーム mikutter 的な用例が観測されています。

薄い本 vol.4 の記事で紹介した「終電逃しツイート via mikutter」に加え、最近では新たに「富士山山頂からのツイート via mikutter」や「アラブ首長国連邦の高級ホテルからツイート via mikutter」などの事例がありました。

また、薄い本 vol.2 および vol.4 の記事でも紹介している「様々なマシンをサポートする NetBSD を使ったあらゆるマシンで mikutter」という流れでは、従来のドリームキャストや W-ZERO3 に加え、初代 Pentium 90MHz PC での動作検証や、20 年以上前のワークステーション LUNA (クロック 20MHz、メモリ 16MB) で mikutter デーモンモードによるテキストによるタイムラインデモ展示などが新たに行われました。NetBSD 以外では、2013 年 11 月の OSC 福岡および OSC 大分では Debian による「kobo で mikutter」というデモが敢行されていました。



LUNA (MC68030 20MHz/16MB) による mikutter デモ

もはや Twitter クライアントとしての mikutter というよりも「そこに mikutter (あるいは謎マシンが) あるから」という理由以外の何者でもないエクストリーム mikutter ですが、この現象は Twitter というコミュニケーションツールにプラスアルファの魅力をもたらしてくれる mikutter ならではといえるでしょう。

9. mikutter 開発協力

ここまでに挙げた mikutter の~~ておくれ~~例使用例は配布されている mikutter を使うだけのものでした。しかし、オープンソースとして開発されている mikutter のもう一つの楽しみ方として、mikutter の開発そのものに協力する、ということが挙げられます。

「開発協力」と聞くと何やら難しくて高度な技術が必要なように聞こえるかもしれませんが、難しい話ばかりではありません。もちろん、過去の薄い本に掲載されているような mikutter のソースコードの内部構造やプラグイン実装といったプログラミング的な濃い話もありますが、それ以外にもこの薄い本 vol.5 の @2GMon さんの Arch Linux 向け mikutter パッケージ作成や、薄い本 vol.4 掲載の @Phenomer さんによる mikutter LiveDVD の作成といった例もあります。こういった内容はこの mikutter の薄い本がもっとも得意としている部分ですので、腕に覚えのある人は薄い本のバックナンバーを片手に新たなプラグイン作成や mikutter 本体の改良、mikutter のディストリビューション作成などに取り組んでみましょう。

また、もっとも身近にできる協力として mikutter のバグレポート送信があります。不幸にして mikutter がクラッシュして終了した場合、その次の起動で mikutter が

mikutter が突然終了してしまったみたいで
ヽ('ω')ノ三ヽ('ω')ノもうしわけねぇもうしわけねぇ

というウインドウを出してバグレポートを送信するかどうか聞いてくる場合があります。この場合、一つのバグレポートが直接 mikutter の問題の解決に役に立つわけではありませんが、どういう現象がどういう環境でどれくらい起きているのかという情報が集まることで、問題が mikutter 側にあるものなのか Twitter サーバー側の不具合によるものか、あるいは特定の OS や環境だけで発生するものなのか等々の分析を @toshi_a さん側で行うことができます。ですので、明らかに mikutter の不具合でないとわかっている場合（自作プラグインのデバッグ中など）を除けばバグレポートは積極的に送信するようにしましょう。

mikutter がクラッシュするような場合はまだわかりやすいのですが、それ以外にも mikutter の不具合（のように見える現象）が発生することがあります。よくありがちなのは「ツイートを取りこぼす」「リアルタイム更新がされない」といった接続周りの問題です。これらの現象は普段が快適な mikutter だけに体感できる不具合として報告が上がりやすい割には原因の特定が難しいという傾向があります。先に書いたように Twitter サーバー側もちょくちょく不具合を発生させることがある上 mikutter は様々な OS およびディストリビューションで動作すること、mikutter 自身のバージョンも日々更新されていることなどから、これらの不具合発生時は各自の使用環境と不具合発生時の詳細な状況を報告することが必要です。

ちょうど、この薄い本 vol.5 の原稿締切日直前の 11 月 22～23 日に Twitter サーバー側の仕様変更(?)が原因と思われる UserStream の接続不具合が発生していたのですが、その際の mikutter ユーザーからの不具合発生状況報告と分析結果連絡が適切に行われた結果、無事 @toshi_a さん側で問題を確認して修正版がリリースされたという事例がありましたので、そこでの一連のツイートを引用してみます。

UserStream 切断問題調査隊
（タイムライン同様下から上へと読んでください）

一番下の @toshi_a さんの問題点登録以前にも@cn さんと@Akkiesoft さんから不具合内容の連絡とそれぞれの ruby のバージョン等の情報が報告されていて、その後問題が再現していた@cn さんの解析結果に基づいて@toshi_a さんが内容を検証し、不具合報告を放置してマイクラに明け暮れるフリをしつつも翌日にはしっかりと修正をコミットされている様子がわかるかと思います。

10. mikutter Redmine サイト

「いきなりそんな不具合の解析や報告なんてできないよ～」というユーザーもいるかと思います。そういう方は、上記ツイートでとしぁさんが不具合をメモっていた http://dev.mikutter.hachune.net/projects/mikutter/ の mikutter Redmine のサイトをながめてみるとよいと思います。

Redmine は web ベースのプロジェクト管理ツールで、ソースコードのバージョン管理や問題点管理を一元的に行うツールで、mikutter の開発はこの Redmine を使って管理されています。ページトップには何やら怪しげな QB さんの契約のお誘いが書いてあったりします

が、mikutter の Redmine サイトに対して報告を行う場合には@toshi_a さんに対してメンバー登録申請が必要になるものの、とりあえず外から様子をながめてみるだけであれば申請は不要で、web 経由で各種の mikutter 開発状況の情報を自由に見ることができます。

mikutter Redmine サイト トップページ

Redmine のサイトでは、主に過去の履歴を含むソースコード、各ソースの変更の履歴、そして「チケット」と呼ばれるいわゆる「問題点」の報告メモから構成されています。

ソースコードそのものはバリバリプログラム本体をいじったりデバッグしたりする人向けですが、変更の履歴とチケットについては日本語で記述されているため、特にプログラミングがわからない人でも mikutter のリアルタイムな開発状況として眺めることができます。

日付	作成者	コメント
11-23 16:48	toshi_a 初音	streaming: gzipやdeflateでUserStreamに接 が遅れることがあるので、圧縮しない refs #626
11-23 15:46	toshi_a 初音	streaming: 行の途中でデータが切れた場合、ク クトが無視される refs #626
11-11 20:49	toshi_a 初音	extract: 抽出条件をRubyへ変換して実行する
11-11 16:47	toshi_a 初音	MIKU: Rubyコードを書き出す機能
11-11 00:26	toshi_a 初音	MIKU: 関数とテスト追加
11-10 16:37	toshi_a 初音	チュートリアルの仮のフローを削除

ソースコードレポジトリ リビジョン一覧

ソースコードの変更履歴は@toshi_a さんが mikutter に対して行った修正を過去にさかのぼっていつ、どのファイルを修正したかを含めすべて確認することができます。簡単なコメントもついているので、いつどんな問題が発生し、どれくらいの期間でどうやって解決されたか、また、mikutter に対して変更が行われた日時を観測することで @toshi_a さんが本業でデスマーチに陥っていた季節を感じることができるなど、いろいろな情報を読み取ることができます。

#	トラッカー	ステータス	優先度	題名
626	環境対応	様子見	通常	UserStreamがgzipで転送されている時、JSONオブジェクトの途中でチャンクを切るように変更されたことに対する対応
625	機能	新規	通常	ショートカットキーインポート・エクスポート機能
624	バグ	解決	通常	openimg: タグのスクレイピング方法を修正
620	最適化	新規	通常	プラグインディレクトリの統廃合
618	致命的	新規	通常	Ruby-GNOME2のHEADを持ってくるとマウスポインタをTimeLineに乗せると落ちる
617	バグ	新規	通常	Gemfileの読み込みパスに${MIKUTTER_DIR}/pluginディ

Redmine チケット一覧

チケットは@toshi_a さん自身を含めた mikutter ユーザーから報告された問題点や改善要望、修正パッチなどです。内容は様々で、先ほどのリアルタイム更新切断の問題が記載された#626 のチケットのように報告された詳細情報が記載されたものもあれば、「#564 ダークマターが潜むプラグインの特定」「#319 自分をふぁぼろうとしたら警告ダイアログを出す」といったアヤしさやユーモアを感じさせるチケットもあります。

これら履歴とチケットは日々更新されていくためここでは詳細まで触れませんが、ブラウザで Redmine サイト全体を眺めながら興味を惹かれたものをポチポチとクリックしてみると mikutter に対する新たな発見があるかもしれません。

11. 終わりに

はじめにも書いたように、この 12 月で mikutter も 4 年目を迎えています。これまでの mikutter の開発状況を @toshi_a さんの blog で確認してみると、1 年目は公式 RT や StreamingAPI が登場し、2 年目は t.co による公式短縮 URL とアクティビティ、そして画像投稿対応、そして震災による Twitter 自体の役割の変化やふぁぼのインフレがありました。3 年目はいろいろと物議を醸した API 1.1 および悪名高き Display Requirements の発表があり、mikutter 自身も薄い本の発刊や@toshi_a 自身による OSC 京都でのセミナー開催、そして mikutter 0.2 のリリースがありました。

一方、4 年目の今年を振り返ってみると、API 1.0 の停止というイベントはあったものの、Display Requirements については表面上の動きはなく、Twitter 社の株式上場という一大イベントがあった割にはユーザーから見える変化はあまりありませんでした。Twitter サービスそのものも公式 web 会話表示の青線など小規模にはバタバタしたものの、バルス祭りのようなイベントを除けばクライアントを悩ませるようなトラブルもなく、本文で紹介した原稿締切直前の UserStream 問題が新鮮に思えたほど平穏(?)な 1 年

だったように思えます。

mikutter 自身も@toshi_a さんが自宅の京都を離れ大阪国に長期軟禁されるデスマーチ状態が長かったこともあり、次期開発バージョンの 0.3 の開発は進んでいるとはいえ公式リリースバージョンは 0.2.x のまま 1 年が過ぎようとしています。

@toshi_a さんがておくれていなければ、この薄い本 vol.5 がコミケに並ぶ時期には mikutter 0.3 もリリースされていることと思います。0.3 にはこの薄い本 vol.5 の @toshi_a さんの記事にあるようにマルチアカウント対応などこれまでの mikutter とは違う新たな方向性の機能が盛り込まれています。

私が mikutter を使い始めたのは 2011 年の 4 月ごろですが、それから約 2 年間は Twitter そのものが大きく変化し、また mikutter も Twitter 公式（そしてふぁぼ戦争を始めとする多くのユーザー）にもまれながらも大きく成長していったように思えます。しかし、先に書いたように API 1.0 廃止以降の Twitter は特に大きな変化もなく、Twitter 社上場後の株価も今のところは堅調のようで、近いうちに Twitter が大きく変化することはないと思われます。また、mikutter というクライアントも不足していると思われていた機能はあらかた実装され、一応の完成に近づいて来ているような気がしています。

今までの mikutter 開発の原動力は、多くのユーザーの支援があったのはもちろんですが、一番の要素は開発者である@toshi_a さん自身の「俺はこういうクライアントが作りたいんだ！」という熱い思いにあったことは間違いありません。しかし、Twitter というプラットフォーム自体が成熟期に入りつつある今後は、私たち mikutter ユーザー一同が@toshi_a さん以上に熱い思いを持って mikutter という環境を盛り上げていきたい、そんなことを思って、今回いまさらと思える mikutter の導入からの記事を書かせていただきました。この記事が mikutter の今後の発展の一助となれば幸いです。

AURのパッケージング作業を振り返って

つじもん (@2GMon)

1. はじめに

私がmikutterを使い始めたのは2011年の3月下旬でした。すぐに落ちたり、メモリをバカ食いしたりといったことはありましたが、初めて使った時からmikutterが大好きになり自分が利用する全てのマシンでmikutterを使えるようにしたいと思いました。しかし、当時はパッケージングされておらず、svnからチェックアウトしたり不安定版をダウンロードして、PATHを通すという作業が必要でした。マシンの数だけこのような反復作業をするのが面倒だったのでパッケージングしようと思ったのが始まりです。

現在は私がmikutterを使うマシンは1台になっており、gitで最新版を追いかけているため個人的にパッケージは必要なくなっています。しかし、パッケージの更新を楽しみにしておられる方がいるようですし、mikutterを初めて使う方のハードルを下げたいと思いパッケージ作業を続けています。AURのパッケージでmikutterを使い始めた方の中からコミュニティに貢献してくださる方が現れると嬉しいです。

本記事では、AURのパッケージについて紹介していきたいと思います。

2. AURとは

AURとはArch User Repositoryの略で、公式のリポジトリに含まれていないパッケージをユーザーが投稿できるリポジトリです。4万個以上のパッケージがユーザーによってメンテナンスされています。Arch Linuxのパッケージは PKGBUILD というシェルスクリプトを書くだけでよいので多くのユーザーがパッケージを投稿しており、mikutterのパッケージも簡単なシェルスクリプトが書かれたファイルだけで構成されています。

最近はmikutterが安定しているため、パッケージング作業はPKGBUILD内のバージョンとハッシュ値を書き換えるだけになっています。

3. mikutterパッケージ誕生

AURのパッケージにはVCSからチェックアウトしてインストールするパッケージも多くあり、mikutterのパッケージも最初はsvnからチェックアウトするかどうかで悩みました。しかし、当時のmikutterは不安定だったので人柱になる勇気はなく、手元の環境で動作した不安定版ver. 0.0.2.12をパッケージングすることに決めました。

ただし、ver. 0.0.2.12がそのまま動作したわけではありませんでした。Ruby 1.9.2からLOAD_PATHにカレントディレクトリが含まれなくなっていたので起動せず、パッケージ側で1行だけのパッチを当ててLOAD_PATHを追加していました。その後、リビジョン831bb034でLOAD_PATHの問題は解消されたので、パッチは削除されています。

AURのパッケージではダウンロードしてきたファイルを/opt/mikutterに展開しています。FHSで/optディレクトリはアプリケーションごとにサブディレクトリで分けて格納することを想定されているので、/usr/localよりも自然だと考えたからです。しかし、このままでは起動コマンドがruby /opt/mikutter/mikutter.rbとなってしまい、毎回打つのが面倒です。そこで、このコマンドを書いた起動スクリプトを用意しておき/usr/binにコピーするようにしました。FHSに従うのであれば/opt/mikutter/binディレクトリを用意してPATHを通すのが良いのでしょうが、あまり環境変数を変更したくなかったので現在のファイル構成になっています。

mikutterが依存しているgemは全てAUR上に存在したので全てを記述すれば問題なかったのですが、依

存するパッケージの記述が多くなるとメンテナンスの手間が掛かりそうだったのでシンプルな記述を目指しました。幸いなことに、それらの gem は ruby-gtk2 パッケージに必要とされていたので mikutter の PKGBUILD には ruby-gtk2 を記述するだけで済みました。

mikutter 本体は相変わらず不安定でしたが、0.0.3, 0.0.4, 0.1 と順調にバージョンが上がっていきました。複数バージョンが並行して存在する期間はパッケージでどちらのバージョンを提供するか悩んだりしましたが、どちらのバージョンでも不安定なことに変わりはなかったのでバージョンが高い方を提供していました。その間、編集が必要になったのは PKGBUILD の標準に従っていなかったライセンス表記だけで、それ以外の箇所を編集する必要がありませんでした。

4. パッケージ第 2 世代

OSC 2012 Kansai@Kyoto で mikutter 開発者のとしぁさんと少しだけお話をさせていただき、いつの間にか AUR に登録されていてビックリした、自分でやろうとは思わないから嬉しい、みたいな話を聞いたような気がします。そこで、久しぶりに PKGBUILD を見直そうと思い第 2 世代の PKGBUILD が登場しました。第 2 世代と言っていますが、変更したのは起動スクリプトの配置方法のみで、それ以外は初代のパッケージと同じです。

mikutter 0.1.1.814 までは最初に書いた PKGBUILD と起動スクリプトの組み合わせで提供していましたが、起動スクリプトを別に用意するのはメンテナンスに手間がかかると考えて PKGBUILD に統合しました。統合するためにヒアドキュメントを利用して PKGBUILD 内で起動スクリプトを生成するようにしました。その後、mikutter のバージョンは 0.2 になりましたが、この変更以降も PKGBUILD を変更しなくてもパッケージの生成ができていました。しかし、起動スクリプト生成時に変数展開されてしまっていて起動スクリプトにオプションを渡しても無視される状態のまま半年ほど放置してしまっていました。AUR のコメント欄での指摘で問題に気づいたため mikutter 0.2.1.1141 で修正しています。

5. パッケージ第 3 世代

パッケージ第 2 世代でも問題なくパッケージング作業はできていたのですが、mikutter 0.2.2.1297 あたりのバージョンは、Display Requirements をアレするプラグインをインストールしていない場合は起動できないという問題がありました。初めて導入する方のハードルを下げるというパッケージ提供目的もあったので、インストールやアップデートした後に Display Requirements をアレする方法を提示するように変更しました。mikutter 0.2.2.1318 のパッケージから導入されているこの変更によって、起動できないという問題が解決した後も、初めてインストールするユーザーに対して利便性を向上させることができるのではないかと考えています。

mikutter にはパッケージ提供者がプラグインを予めインストールできる plugin ディレクトリが用意されていたため、Display Requirements をアレするプラグインをインストールしてパッケージを提供することを考えたのですが、規約に違反したものを提供するのはマズイと思いユーザーに手段を提示するだけの現在の形に落ち着いています。

6. mikutter パッケージのこれから

ここまでで、mikutter をパッケージングするためにどのような作業を行ってきたのか説明しました。本章では、これからの mikutter のパッケージングについて現在考えていることを少しだけ説明しようと思います。

パッケージング作業自体は今までと同じくバージョンを書き換えるだけです。しかし、現在ダウンロードしてきたファイルのハッシュ値に脆弱性が発見されている MD5 を利用しています。そこで、SHA-2 の利用を検討しています。これはすぐにでも対応できるのですが mikutter 0.3 のリリースを待ってから変更するつもりです。

7. おまけ

AUR のパッケージは更新が容易なため、不安定版のリリース後、かなり早い段階でパッケージの更新ができていると思っています。そこで、手元に残っている mikutter 0.0.3.5 以降のバージョンについて mikutter 開発日記の投稿時刻とパッケージの作成時刻を比較してみたところ意外と早いわけではないことがわかりました。たまに mikutter 開発日記更新後 10 分以内に更新があるぐらいで、残りのほとんどは半日から 1 日経過後の更新でした。特別早かった時だけが印象に残っていて、他の場合のことは全然残っていないなんて、人間の感覚はあまりあてになりませんね。

バージョン	開発日記投稿時刻	パッケージ作成時刻	時間差
0.0.3.5	2011/5/23 0:10	2011/5/23 0:14	4min
0.0.3.6	2011/5/28 23:14	2011/5/29 0:23	69min
0.0.3.7	2011/6/5 14:59	2011/6/5 23:10	491min
0.0.3.9	2011/6/19 0:56	2011/6/19 13:09	733min
0.0.3.10	2011/7/3 18:02	2011/7/3 21:27	205min
0.0.3.11	2011/7/9 12:47	2011/7/17 12:06	11479min
0.0.3.12	2011/7/16 23:06	2011/7/17 12:06	780min
0.0.3.13	2011/7/23 13:15	2011/7/23 15:51	156min
0.0.3.14	2011/7/30 1:10	2011/7/31 9:27	1937min
0.0.4.456	2011/8/7 21:23	2011/8/8 11:40	857min
0.0.4.460	2011/8/14 16:18	2011/8/18 9:14	8216min
0.0.4.472	2011/8/21 0:09	2011/8/21 12:17	728min
0.0.4.485	2011/8/27 16:38	2011/8/29 9:22	2444min
0.0.4.505	2011/9/12 23:48	2011/9/17 2:38	5930min
0.0.4.514	2011/9/21 2:25	2011/9/21 15:12	827min
0.0.4.522	2011/10/2 1:40	2011/10/2 11:26	586min
0.0.4.534	2011/10/10 16:27	2011/10/11 12:31	1204min
0.0.4.549	2011/10/17 1:17	2011/10/17 12:38	681min
0.0.4.556	2011/10/23 15:19	2011/10/23 19:17	238min
0.0.4.571	2011/11/4 0:07	2011/11/4 8:53	526min
0.1.0.580	2011/11/14 2:33	2011/11/114 12:37	604min
0.1.0.592	2011/11/19 18:43	2011/11/19 20:17	94min
0.1.0.603	2011/11/27 4:42	2011/11/27 23:04	1102min
0.1.0.609	2011/12/11 2:06	2011/12/12 10:30	1944min
0.1.0.615	2011/12/17 23:27	2011/12/19 13:25	2282min
0.1.0.622	2011/12/26 2:39	2011/12/26 12:29	590min
0.1.0.628	2012/1/3 2:43	2012/1/5 11:43	3420min
0.1.0.638	2012/1/10 0:25	2012/1/10 14:14	829min
0.1.0.642	2012/1/15 5:02	2012/1/15 10:42	340min
0.1.0.658	2012/1/30 21:26	2012/2/1 10:51	2245min
0.1.0.690	2012/2/18 22:35	2012/2/18 23:11	36min
0.1.0.697	2012/2/26 21:53	2012/2/26 22:37	44min
0.1.0.710	2012/3/2 1:25	2012/03/11 13:14	13669min

バージョン	開発日記投稿時刻	パッケージ作成時刻	時間差
0.1.0.723	2012/3/24 0:59	2012/3/24 01:02	3min
0.1.0.749	2012/4/21 18:18	2012/4/21 20:39	139min
0.1.0.772	2012/5/1 14:25	2012/5/6 11:10	7005min
0.1.1.780	2012/5/8 2:04	2012/5/8 11:50	586min
0.1.1.790	2012/5/13 22:43	2012/5/13 22:58	15min
0.1.1.799	2012/6/9 13:08	2012/6/9 17:50	282min
0.1.1.810	2012/6/17 23:07	2012/6/17 23:44	37min
0.1.1.814	2012/7/8 19:40	2012/7/8 21:09	89min
0.1.1.863	2012/9/4 22:15	2012/9/4 22:26	11min
0.2.0.1045	2012/10/5 9:02	2012/10/5 11:29	147min
0.2.0.1051	2012/10/15 23:17	2012/10/15 23:22	5min
0.2.0.1054	2012/11/7 0:55	2012/11/7 00:58	3min
0.2.0.1064	2012/11/23 22:24	2012/11/23 22:27	3min
0.2.0.1080	2012/12/10 0:13	2012/12/10 01:10	57min
0.2.0.1089	2012/12/15 18:05	2012/12/16 01:44	459min
0.2.1.1112	2013/1/4 21:06	2013/1/6 17:09	2643min
0.2.1.1117	2013/1/9 22:35	2013/1/10 17:09	1114min
0.2.1.1119	2013/1/12 18:37	2013/1/12 20:16	99min
0.2.1.1125	2013/1/26 19:28	2013/1/27 13:42	1214min
0.2.1.1127	2013/2/5 0:40	2013/2/5 23:01	1341min
0.2.1.1130	2013/2/22 22:01	2013/2/23 02:18	257min
0.2.1.1132	2013/3/10 12:07	2013/03/10 12:28	21min
0.2.1.1137	2013/3/22 0:04	2013/03/24 19:51	4067min
0.2.1.1141	2013/3/27 1:50	2013/03/27 11:46	596min
0.2.2.1211	2013/4/20 0:16	2013/4/20 13:11	775min
0.2.2.1225	2013/5/3 0:41	2013/5/3 09:42	541min
0.2.2.1230	2013/5/8 22:00	2013/5/9 20:40	1360min
0.2.2.1264	2013/6/12 0:13	2013/6/12 22:21	1328min
0.2.2.1297	2013/7/14 23:54	2013/7/15 00:17	23min
0.2.2.1318	2013/8/4 20:50	2013/8/4 22:29	99min
0.2.2.1328	2013/8/14 8:54	2013/8/14 21:20	746min
0.2.2.1373	2013/9/28 14:13	2013/9/28 17:11	237min
0.2.2.1410	2013/11/4 23:21	2013/11/4 23:46	25min

2014 年の mikutter plugin 管理方法

@osa_k

この記事では、mikutter plugin の管理方法について現在の状況を整理しつつ、これからのありかたについて考えていきます。

1 mikutter plugin の仕組み

この本を読んでいる人にはもはや説明するまでもないかもしれませんが、mikutter plugin の仕組みについておさらいします。

mikutter plugin は Ruby スクリプトであり、同じく Ruby 製である mikutter が実行時に動的にこれらのスクリプトを読み込むことで動作します。以下に示すコードは、mikutter plugin の一例です[*1]。

```
#-*- coding: utf-8 -*-

Plugin.create :eject do
  command(:eject,
          name: 'eject',
          condition: lambda{|opt| true},
          visible: false,
          icon: "#{File.dirname(__FILE__)}/cd_eject.png",
          role: :window) do |opt|
    Thread.new {
      system("eject -T #{UserConfig[:eject_device]}")
    }
  end

  UserConfig[:eject_device] ||= ''
  settings 'eject' do
    fileselect 'Device', :eject_device
  end
end
```

通常、mikutter plugin は `Plugin.create` から始まります。この中では `command` や `settings` といった DSL が定義されており、これらを呼び出すことで mikutter のコンテキストメニューや設定画面へ干渉することができます。その他にも、mikutter が Tweet を受信したり、ユーザがタブを切り替えたりするときなど要所要所でイベントが発生するようになっており、それぞれのイベントに対してハンドラを登録することもできます。

また、mikutter plugin は spec ファイルをもつことができます。spec ファイルはプラグインの情報を記述してある

[*1] このコードはボタンを押すと CD トレイを排出するだけのプラグインです。ジョークです。

YAML 形式のファイルで、以下のようなものです。

```
---
slug: :eject
depends:
  mikutter: '0.2'
  plugin:
  - command
version: '1.0'
author: osa_k
name: Eject
description: mikutter 使ってるとよく CD-ROM を取り出したくなりますよね．ならないとしたら，今すぐこのプラグインの使用をやめて病院に行って下さい．
repository: git://git.github.com/osak/mikutter-eject.git
```

プラグインの名称、作者、説明、依存関係などが書かれています。このファイルはプラグインと同時に mikutter が読み込み、依存関係が満たされないプラグインを無効化したり、名前や説明を参照するためのイベントを定義したりします。

2 現在の mikutter plugin 管理

2.1 Git

現在、mikutter plugin の管理方法として主流なのは Git を使う方法です。mikutter に最初からバンドルされていない、いわゆるサードパーティのプラグインはほとんどが GitHub で公開されており、ユーザは手作業で `git clone` することでプラグインを導入します。

```
$ git clone git://github.com/osak/mikutter-eject.git ~/.mikutter/plugin/eject
```

この方法は公開する側にとっては手軽で良いのですが、ユーザが Git を使えること、Linux のディレクトリ構造等についてある程度理解していることなどの前提が必要であり、少々複雑であるという問題があります。特に Windows は Git とあまり相性がよくないため、mikutter を使いたいだけの一般ユーザにとってはかなりの負担となります（Windows は mikutter の動作保証範囲外ですが、実用的に動かすことは可能ですし、実際動くのであれば使いたいという人は少なからずいると思います）。また、プラグインの管理はディレクトリベースとなり、1 エディタである Vim ですらプラグインマネージャをもっている[*2]現代において、いささか時代遅れな感じは否めません。

2.2 みっくストア

上記の問題を解決するため、@toshi_a さんによって開発された mikutter plugin がみっくストア（https://github.com/toshia/mikustore）です（図 1）。これは mikutter plugin を管理するための mikutter plugin で、GUI 上でプラグインの導入状況を見たり、ボタンひとつでインストール・アンインストールをしたりすることが可能です。

このプラグインで煩雑なシェル操作から解き放たれて万々歳！……と言いたいところですが、実はみっくストアでもまだ解決できていない問題があります。図 1 で右ペイン左側に出ているのは導入可能な mikutter plugin の一覧で

[*2] Vundle や NeoBundle というプラグインが有名です。

図 1　みっくストアプラグイン

すが、この一覧を管理しているのもまた mikutter plugin なのです。上の方に出ている `toshi_a's store` というのがその mikutter plugin のひとつで、特定のイベントに反応して、管理しているプラグイン一覧を spec の形で返すようになっています。みっくストアはこの情報を使っています。

```
# -*- coding: utf-8 -*-

Plugin.create :toshi_a_store do
  filter_mikustore_plugins do |plugins|
    Dir[File.expand_path(File.join(File.dirname(__FILE__),
                                   "plugins", "*"))].each{ |package_file|
      plugins << YAML.load_file(package_file).symbolize
    }
    [plugins]
  end
end
```

```
$ ls toshi_a_store/plugins
display_requirements  nested_quote  open_favstar  sub_parts_client  tl_security
```

この方法の問題点は 2 つあります。ひとつは、spec ファイルが store とプラグイン本体で二重管理されることです。このため、プラグインを更新するとき同時に store 側の spec ファイルも更新し、常に情報が最新になるように注意する必要があり、mikutter plugin 作者にとっては手間が増えてしまいます。

もうひとつの問題は、store 自体の導入にはどうしても Git をシェルから叩く必要があることです。今は個人個人で store を管理し、ユーザが適宜 store を導入していくという形を取っています。たとえば@penguin2716 さんの pengui2716_store（https://github.com/penguin2716/penguin2716_store）や、@osa_k の osa_k_store（https://github.com/osa_k/osa_k_store）といった store があります。これらの store そのものの情報を管理する store は無いため、どうしてもユーザが直接取りに行く必要が生じてしまいます。

また、みっくストアは内部的に `system` で `git` コマンドを呼び出す実装になっているため、Windows で動かすの

がやはり難しいという問題もあります（ただし、`git` コマンドを使わずに Git リポジトリを操作することができる `grit` という gem があるので、`git` コマンドへの依存は解消できるのではないかと考えています）。

3 これからの mikutter plugin 管理

先に挙げた問題は、mikutter plugin を一括管理するようなサーバがあれば解決することができます。つまり、store をユーザごとに管理するのではなく、管理サーバ 1 つで store1 つという構造にして、store の持っているプラグイン情報はサーバ側で自動生成するようにするのです。

3.1 Mikuregator

上記のアプローチで、@ichigotake さんが Mikuregator（http://mikuregator.ichigotake.net/）というサービスを開発されています。Mikuregator は GitHub をクロールして mikutter plugin っぽいリポジトリを一覧で表示してくれるサービスで、この記事を書いている 12/8 現在では store の自動生成は対応していませんが、GitHub の Issues を見ると、今後の機能追加で実装する予定があるようです。

3.2 Mikkurepos

リポジトリとして GitHub を使うのではなく、各種プログラミング言語や Linux ディストリビューションのように、独自のリポジトリサーバを用意するという方法もあります。

プラグイン方式を採用している有名な Ruby アプリケーションとしては pry や fluentd があり、これらのアプリケーションでは rubygems を使ってプラグインを管理しています。しかし、mikutter は@toshi_a さんの意向により本体を rubygems に置かないということになっているため、プラグインを rubygems で配布するのは少しはばかられる所があります。

そこで現在、自分はこのアプローチでプラグイン管理を実現するべく、Mikkurepos というサービスを開発しています。イメージとしては Arch Linux の AUR のような感じで、アップロードされた mikutter plugin をサーバ側で分類し、リストをまとめて提供するようなものを考えています。

3.3 他の方法

さらに別のアプローチとして、URL を指定するだけで mikutter plugin をインストールできるような機能をみっくストアに付加するというものも考えられます。この方法だと、きちんとしたリポジトリの形で公開されていない、Gist に貼ってあるだけのようなプラグインでも導入できるという利点があります。

4 まとめ

現在開発中[*3]の mikutter0.3 では様々な新機能が追加され、mikutter plugin 開発を支援するような機能もあります。また、vol.4 で紹介したデーモンモードのように、プラグイン機構を前面に押し出した使い方も開拓されてきています。こういった状況の中で、どうせ ておくれ しか使わないからといって曖昧にされてきた、ユーザ視点に立った mikutter plugin 管理手法は重要性を増していくと考えられます。

プラグイン機構自体は mikutter の設計思想レベルで存在している概念ですが、その管理方法についてはまだ試行錯誤の段階にあると思っています。どういう管理方法が良いのかを知るためには実際にいくつか実装して、うまく機能するものを拾っていく必要があるでしょう。

[*3] この記事が出る頃にはもうリリースされているかもしれません。

複数のTwitterアカウントを取り扱うために

@toshi_a

mikutter 0.3からは、一つのmikutterに二つ以上のアカウントで同時にログインできるようになります。これを普通のTwitterクライアントは「マルチアカウント機能」と呼びますが、mikutterでは他のそれとは違ったコンセプトを元に設計されています。今回は、mikutterのマルチサービス機能の考え方を、プラグイン開発者向けに説明したいと思います。もちろん、プラグインを作らないユーザにとっても、興味深い内容だと思います。

1 抽象サービスレイヤの歴史

mikutterはもともと、コンピュータを監視するbotを作成したところからスタートしました。Twitterのbotではありません。私は以前、メールでアラートを受け取っていましたが、忙しくなると大量のメールをチェックするのをつい怠ってしまい、それらを見逃してしまっていました。そこで、送るサービスを変更できたら便利なんじゃないかと思い、送信先を変更できる抽象レイヤーを書いて、自分がメールよりよく見ているTwitterに対してアラートを送るようにしました。 その後程なく、このプラグインシステムを使ってTwitterクライアントプラグインを書こうと考えるに至ったのです。そういうわけですから、そもそもmikutterがTwitterの方向を向いたのは、意外と後になってからの話だったのです。

1.1 「死なないために」封印された機能

mikutterの進化は常に綱渡りでした。決して順風満帆ではなく、可能な限りできることを諦めて、墜落しないように前に進むことだけを考えていたのです。最初の頃のmikutterは「全てのことができるように」という目標を掲げていたにも関わらず、驚くほど何もできませんでした。いろいろなことを諦めてここまで来ましたが、いつの間にか振り返ってみると、必要最低限の機能は揃ったようです。足りない部分も多くはプラグインが補ってくれるようになりました。 「サービスの抽象化」はこの過程で切り捨てた機能のひとつで、いくつかの重要な部分はTwitter前提になってしまいました。 結果として、Serviceクラスはもともとsnsとそのアカウントのペアだったのに、SNS側がTwitterに固定されてしまい、事実上 **「Service = アカウントのインスタンス」** になりました。

1.2 マルチサービスの遺産

抽象サービスレイヤーは一度捨てられましたが、実はとても身近なところにその面影が遺っています。例えば、プラグイン開発を勉強し始めて程なく見るであろう、ツイートを投稿するコードを見てください。

```
Service.primary.post(message: "hello mikutter!")
```

Service.primary は、現在の **Service** のインスタンスを返すものです。 **Service**

とは、上述の通りアカウントのことです。実は、次のようにすればそれが指すアカウントを得ることができます（「# =>」の後ろは、その行のそれより前に書いてあるコードの戻り値です）。皆さんも、自分のmikutterで ALT + xを押してコンソールを開き、次のコードを実行してみてください。

```
Service.primary.user # => "toshi_a"
```

あなたのアカウントは、今までのmikutterではシングルトン（のようなもの）で保持されていたのです。でもRubyでよく見るシングルトンとはメソッド名が違います。あたかも2つ以上ありそうな名前です。実は、Service#all というメソッドも有ります。

```
Service.all # => #<Set: {#<Service toshi_a>}>
```

これは、toshi_aというサービスだけが入っているという意味です。これがあなたのmikutterで実行できたということが重要です。0.2系で、既にこのメソッドはある、つまり複数サービスを扱うことを想定していたのです。 尤も、0.3ではallというメソッドは削除されるなど、実際に複数のアカウントを取り扱うための変更はされましたが、多くのプラグインはマルチアカウントに対応したmikutterのために一行もコードを書き足す必要はないのです。 もちろん、いわゆる「マルチアカウント」機能は、この抽象サービスレイヤーを利用して実装されています。

2 mikutterのマルチアカウントの考え方

プラグインを作成するにあたり、mikutterのマルチアカウント機能の考え方について理解するべきです。

2.1 TLが混ざってる

mikutterでアカウントを追加して見た目に起こる変化は、ツイート入力欄の左に現在のアカウントを現すアイコンが表示されるだけです。そしてなんと、アカウントを切り替えても表示は何も変わりません。昨今のスマートフォン向けTwitterクライアントのマルチアカウントに慣れていると、home timelineが「現在のアカウント」の物にならないのかと思われる人が多いと思いますが、なりません。 home timelineは、「 **update** イベントに渡されたツイートを全て表示するもの」簡単に言えば、「全アカウントの home timeline をごちゃまぜにしたもの」ということになります。これは、Mentionも同じです。Saved SearchやListも全アカウントのものが使用されます。

2.2 今のアカウント

Service.primary で取得できるのは「現在の」アカウントです。操作を一つのアカウントに絞るべき時に、これが使われます。例えばツイートを投稿する時です。リストは全アカウントのものが平等に扱われますが、リストを作成する時はプライマリアカウントのものとして作成されます。ツイートをお気に入りに追加した場合はどうなると思いますか？

2.3 便利？不便？

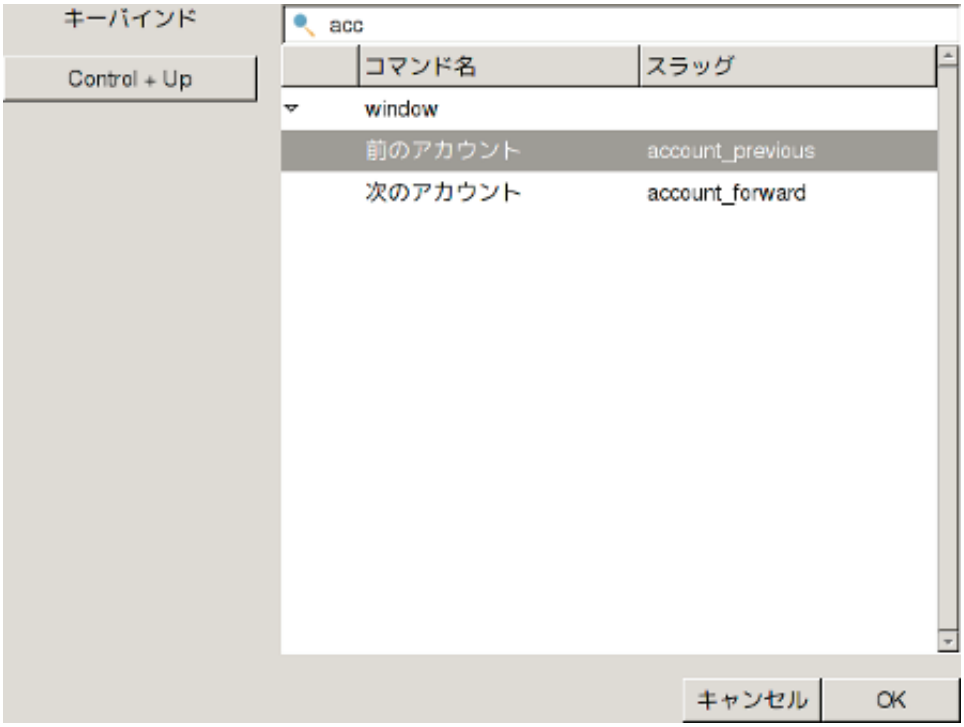
この仕様を何人かに話したら気に入っていただけたので驚きました。私はこれを好ましく思わない人が多いと思っていたからです。恐らくこの仕様は規制回避目的でアカウントを複数用意する人には喜ばしいことですが、実生活とネット上を分けたいという人には歓迎されないでしょう。しかし、こんな仕様でユーザは一喜一憂する必要はありません。カスタマイズする方法があるのです。そのひとつが extract プラグイン、抽出タブです。

抽出タブは生まれてから今まで不遇のプラグインでした。かなり込み入った条件設定ができる代わりに、取っつきにくいUIを持ってしまい、難しいものであるという印象を与えてしまいました。 そもそも上級ユーザ向けの機能であったことも災いし、標準プラグインでありながらあまりユーザがいないと考えられています。 しかし、0.3では extract のデータソース機能を拡張し、「各アカウント又は全アカウント」の「Home Timeline、Mention等」を指定することができるようになります。また、データソースはミックスできます。つまり、アカウントAとBを混ぜたタイムラインと、Cだけのタイムライン、Mentionsは全部一緒にしたもの、という使い方もやろうと思えばできます。 勿論この範疇に収まらないことでも、簡潔なマルチアカウント用のAPIを使用して、プラグインを作成できるでしょう。

3 マルチアカウントに対応するべきプラグインとその方法

多くのプラグインはそのままで動作します。しかし、複数アカウントがあることで意味合いが変わるプラグインや、複数アカウントがあることで追加機能を提供できるプラグインが存在するでしょう。実際にいくつかの標準プラグインの事例を紹介します。

3.1 大幅な変更の例 change_account



change_account は、プライマリアカウントを変更する機能と、アカウントを認証して追加する機能を持っています。これが大幅に変更されるのは言うまでもありません。

3.1.1 前/次のアカウントに切り替え

mikutterコマンドで、アカウントを切り替えることができます。コマンドの定義を見てみましょう。

```
command(:account_forward,
        name: _('次のアカウント'),
        condition: lambda{ |opt| Service.instances.size >= 2 },
        visible: true,
        role: :window) do |opt|
  index = Service.instances.index(Service.primary)
  if index
    Service.set_primary(Service.instances[(index + 1) % Service.instances.size])
  elsif not Service.instances.empty?
    Service.set_primary(Service.instances.first) end
end
```

Service.instances が、0.2系の **Service.all** と同じような意味のメソッドですが、これの戻り値が Set ではなく Array になりました。理由は順番があったほうが何かと便利だからです。次のアカウントという言い方も、順序あってのものですね。複数のアカウントがある場合、 instances はそれらを決まった順序の配列で返します。 **Service.primary** で取得できるプライマリサービス（カレントサービス）は、あくまで現在選択されている Service を表現するだけであって、選択されていない Service にたいして処理を行っても構いません。 また、Service関連では、次のようなイベントが追加されています。

名前	引数	意味
primary_service_changed	Service	プライマリサービスが変更された
service_registered	Service	引数のアカウントが新しく追加された
service_destroyed	Service	引数のアカウントがmikutterから削除された

このイベントを使って、アカウントが増減したり切り替えられた時に受動的に動作するプラグインを作成できます。例えば gtk プラグインは、このイベントを受け取って、ウィンドウの右上に現在のプライマリアカウントのアイコンを表示します。ただし、アカウントが2つ未満ならアイコンは表示しません。

自分のアイコンがたくさん

3.2 クラッシュする例 streaming

streaming プラグインは、 UserStream に接続するプラグインです。初めて mikutter を起動したユーザは、アカウント数が0なので、 **Service.primary** は **nil** を返します。これは0.2以前ではありえなかったことです。

```
Plugin.create :streaming do
  streamers = {}                # service_id => ParmaStreamer
  Delayer.new {
    Service.each{ |service|
      if UserConfig[:realtime_rewind]
        streamers[service.name] ||= Plugin::Streaming::ParmaStreamer.new(service)
(以下略)
```

そもそも、今まではUserStreamのコネクションは1本だったのですが、今回からは登録されているアカウント分コネクションを張ります。もちろんコネクションも複数になってしまったので、単純に変数に格納するのではなくHashに格納し、上では省きましたが **service_registered** イベントなどでアカウントが追加されたら UserStream に接続するようにしています。 **Service.primary** を呼ばなくなったので、nilが返ってくるかもしれないという心配はなくなりました。 Serviceが一つもない時、 **Service.instances** の戻り値は空の配列になります。また **Service.each** はそもそもブロックを呼びません。 **Service.primary** が **nil** を返す可能性については注意しておいてください。

4 おわりに

以上のことは全て予定です。出来てないことも書いているので、0.3を目にする頃には変わっているかもしれません。例えば、UserStream のコネクションは、アカウントごとに有効にするかどうかを設定できるようになるかもしれません。 前回の mikutter薄い本vol.4 からは殆ど何もできませんでした。今回は本当に目立った目立ったことがなかったため、0.3で一番インパクトがあるであろう「マルチアカウント」を取り上げました。それでも、実はこの機能、8月には形になっていたんですが…。 0.3は、マルチアカウントをはじめ、プラグイン開発が更に捗る機能をいくつも用意しています。随分遅くなってしまいましたが、皆さんが mikutter をあなた色に染める一助となれば幸いです。

＜＜後書＞＞

@ch_print いかがでしたでしょうか。よいとしぁけを。

@brsywe 忙しくてなにもできなくてごめんなさい。進捗って Amazon に売っていませんか。

――次号予告と寄稿者募集――

原稿提出期限：6 月 30 日

頒布予定：コミックマーケット 86

なお、mikutter の薄い本別冊創刊号は春先の発行を目指しています。

問合せ先：@brsywe , @ch_print

奥付
発行日：2013 年 12 月 31 日(コミックマーケット 85)
PDF 版公開：2014 年 1 月 11 日
（PDF 初版）
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 Amazon ギフト券のメッセージ欄にどうぞ。

mikutter の薄い本制作委員会ウェブページ
http://home1.tigers-net.com/brsywe/mikutter.html
春頃にウェブページを移転させます。（詳しくは前書参照）

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。

A Happy New Year !!