体験版

うぶんちゅ！

まがじん

ざっぱ～ん♪

vol.4

うぶんちゅ！　まがじん ざっぱ〜ん♪ **vol.4** 体験版

うぶんちゅ！　まがじん ざっぱ〜ん♪ の仲間たち　著

2016-01-30 版　　うぶんちゅ！　まがじん ざっぱ〜ん♪ 編集部 発行

Chap. 1

艦これで学ぶ通信傍受入門

柴田充也

現代の戦争において、より正確な情報をより早く取得し情報優勢を獲得することが何よりも重要になっております。各鎮守府の提督諸氏におかれましても対深海棲艦・対羅針盤をより優位に進めるためにも、平時の間から Twitter・まとめサイト・Pixiv・コミケ等を利用したオシント[*1]やそこに付随する参考画像に基づくイミント[*2]に励まれていることと推察します。本稿では、高度な特殊情報を収集・分析・評価に基づいて一段階上のシギント[*3]を行うために、正規な手順に則ったネットワーク傍受の方法を提案します。

1.1 鎮守府と前線の間の通信手段

各鎮守府における提督の皆さまにおかれましては、部隊編成・出撃・演習・遠征・工廠・入渠と日々艦隊運営に勤しまれていることと思います。作戦計画の立案や季節ごとの艦娘の衣装の選定といった提督自身が行う事務仕事だけでなく、艦娘自身による体調やお風呂の管理、艦隊帰投後に艦娘が作成・提出する作戦報告書の内容確認など、艦隊が大きくなるにつれて増加する書類の量に頭を抱えている提督も多いことでしょう。艦娘自身による作戦報告書の作成については、ただでさえ帰投後の疲労度が高い艦娘にとって大きな負担になっているのではと懸念する声もあります。

一般的に艦娘の疲労度は休息・入渠の他に、鎮守府近海を利用した示威行為、間宮印の甘味、提督とのスキンシップなどによって回復すると言われています。しかしながら個々の効果は艦娘本人の機嫌によってしか判断できません。さらに話がややこしくなる要因として、艦娘の中には素直な感情表現を得意としない子がかなりの数いるのです。

幸い前線への指令などの通信はすべて、一定の規約のもとに電子化されています。しかも驚くべきことにこの通信は暗号化はされていません。よってこの通信を傍受・記録し[*4]、その情報を元に艦娘の疲労度の算出や報告書の自動生成を行い、艦娘の疲労度の管理と負担軽減という一石二鳥を目指そうというのが、本稿の目的となります。

1.1.1 艦これにおける情報伝達

まずは各鎮守府の執務室において提督が作成した指令書が、どのような手続き・経路を経て前線・各部署に届くかを概観していきましょう。

執務室で作成した指令書は、まず最初に秘書艦によって電信で利用しやすい JSON 形式[*5]に変換されます。この JSON データが前線や各部署に HTTP[*6]を用いて送られ、その結果が再び JSON 化されて執務室に届き、秘書艦が解釈し提督に伝えます（図 1.1）。また JSON とは別に画像資料や音声資料が HTTP 経由で添付されることもあります。

[*1] オシント（Open Source INTelligence）：メディアなどの公開情報を分析する諜報活動のこと。

[*2] イミント（IMagery INTelligence）：画像解析によって情報を取得する諜報活動のこと。

[*3] シギント（SIGnals INTelligence）：通信や電子信号を傍受することで情報を取得する諜報活動のこと。

[*4] 提督は通信の当事者なので、今回説明する方法は正確には「傍受」ではなくただの「通信の記録」でしかありません。

[*5] JavaScript Object Notation。元々は JavaScript のオブジェクトの表記法でしたが、現在はさまざまなデータのやり取りで使われています。配列の最後の要素の後ろにカンマが使えないことで有名です。

[*6] HyperText Transfer Protocol。いわゆるウェブサーバーとの通信規約の一つです。現在の主流は HTTP/1.1 で、最近は HTTP/2 への

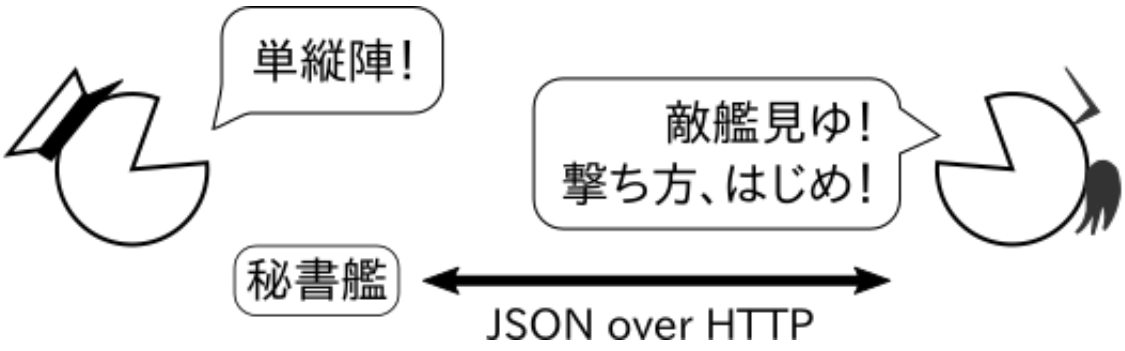

図 1.1　艦娘との連絡は基本的に JSON です。

作戦行動中の通信は、相手が遠隔地ということもあって提督との指示出しは最小限になっています。たとえば提督からは進軍中の陣形を指示するのみで、敵艦隊発見後の交戦開始および攻撃対象の選択や兵装の利用などの交戦中の判断はすべて艦娘が行います。次に通信が発生するのは、戦闘終了後の交戦結果の報告と進軍可否の判断の問い合わせ時になります。このため移動鎮守府などで、たまたま交戦中に一時的に通信範囲外に到達してしまったとしても、指示系統が乱れることはありません。

通信経路に流れた指令内容や作戦結果を把握するだけであれば、ここまでの理解だけでも十分です。しかしながら今回のように通信の傍受を行うのであれば、より通信設備に近い下層での手続きも把握しておいた方が良いでしょう。そこで HTTP 経由で送受信する情報が、より下層ではどのように解釈されているか見ていくことにしましょう。

HTTP に従って符号化された情報を送る前に、まず相手先との接続を確立します。これは相手先を間違えた、もしくは一時的に応答できないことを把握するためです。接続を確立し、情報を送受信し、切断するまでの一連の流れをセッションと呼びます。この接続の確率や再送手続きなどのセッション中の規約をまとめたものが TCP[*7] です。

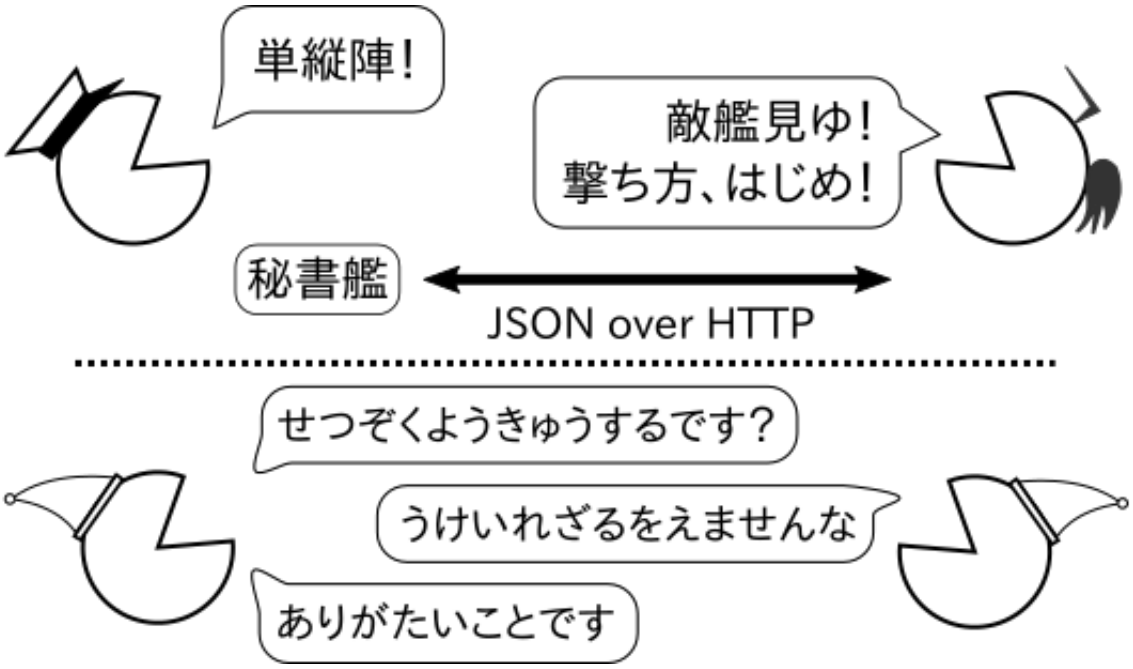

図 1.2　裏では妖精さんが通信を管理しています。

TCP では、まず最初に送り主が「IP アドレスとポート番号」で指定した相手に対して、接続可能か問い合わせます

移行が話題になりつつあります。分母が約 2 倍になっているので、たぶん通信量も半分ぐらいになっているのではないかと思います。

[*7] Transmission Control Protocol。伝送制御プロトコルとも呼ばれます。白ヤギさん黒ヤギさんも、伝送を制御しない UDP（User Datagrap Protocol）ではなく TCP を使っていればあのような悲劇は起こらなかったのでは、と言われています。

(接続要求)。相手先は受け入れ可能であることを送り主に対してを返します（接続要求に対する肯定応答)。最後に送り主は肯定応答を受け取ったことを相手先に肯定応答として連絡することで接続が確立します。ちなみにこれらはいずれも妖精さんが行うため（図 1.2)、提督や秘書艦がその方法を知る必要はありません。この妖精さんは接続の確立だけでなく、後ほど説明する情報の交通整理、迷子の案内、切断処理にいたるまで、通信中のありとあらゆる作業を担当してくれます。逆に言うと妖精さんの調子が悪かったり、妖精さんと通信機器の相性が悪いと通信がまったく行えなくなりますので、くれぐれも妖精さんの機嫌を損ねないようにしてください。といっても、妖精さんは必要以上に酷使しない限り、機嫌が悪くなることはめったにありません。

接続が確立したら、次に妖精さんは指定された情報の送信を行います。この際に大事な作業が「情報の分割」です。大きな情報を一度に送ってしまうと、失敗時の再送に関わる代償が大きくなってしまいます。また経路上では常に大きな情報を送ることができるとは限りません。そこで経路の最小単位にあわせて「パケット」という形で情報を小分けした上で送ることになっています。パケットに小分けにする際に、連番（シーケンス番号）を割り振っておきます。パケットを受信した相手は連番かどうか確認することで、パケットの欠損を認識できるわけです。

妖精さんは小分けしたパケットを「鯖」に載せて相手先に送り出します（図 1.3)。その際に、鯖には相手先の住所だけでなく送信元の住所も付けておきます。妖精さんによって送り出された鯖は、相手先に向かってひたすら泳ぎ続けます。とはいえ相手先の住所だけでは相手にたどり着くまでの経路がわからないため、ところどころで具体的な道順を現地の人に尋ねながら辿っていきます。その際に「道を尋ねた場所」と「次に道を尋ねる場所」を記録しておきます。

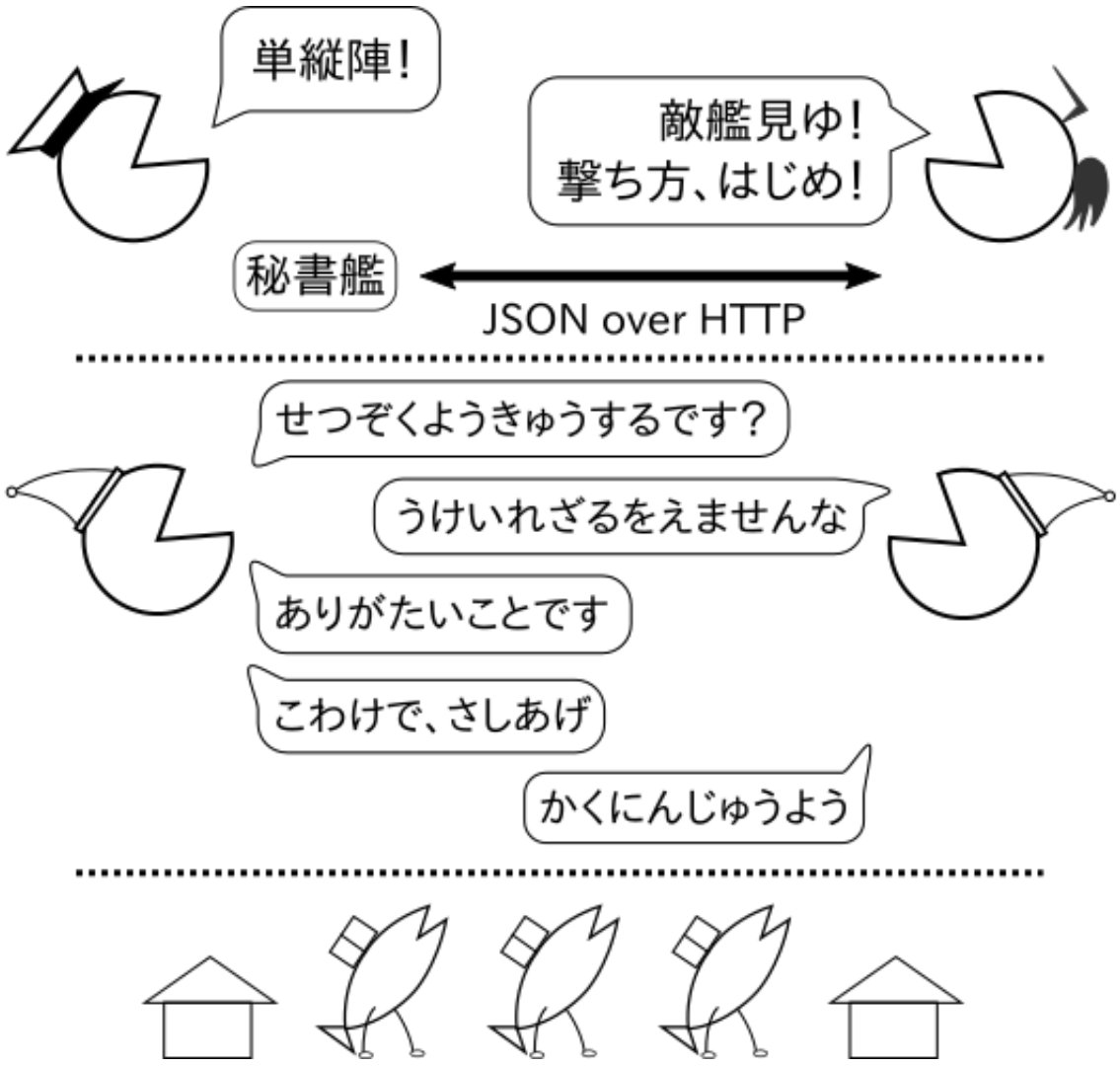

図 1.3　分割されたパケットは鯖が運びます。

無事に相手先にたどりついたら、現地の妖精さんが鯖からパケットを取り出します。パケットの番号を確認して、もし前回受け取ったパケットと順番が異なるようなら送信元に再送要求を送ります。最終的にパケットがすべて揃えば、一つの情報にまとめなおして、秘書艦に手渡します。

その後、送信元に受信確認パケットを鯖に乗せて送り返します。受信確認を受け取った送信元は、他に情報を送る必要がなければ切断パケット（FIN パケット）を送って、セッションを終了します。

ちなみにこの経路は魚だらけなせいか、たまに猫が紛れ込みます。特に鯖の量が増えると、それを狙って猫がたくさんやってきます。猫が鯖を食べるとパケットの欠損が発生し、その再送処理によって負荷があがっていき、結果的に通信がまったくできなくなってしまうのです。そこで、高負荷時には妖精さんが直接、経路に詰まった猫を引っこ抜きます[*8]。通信問題が発生した時は、必要以上に妖精さんの作業を増やさないためにも、大人しく待つようにしましょう。

1.1.2 tcpdump による通信の傍受

通信手続きの概要がわかったので、具体的に傍受を行ってみましょう。ここからは鎮守府の環境が Ubuntu 上に構築されていることを前提に説明します。Windows や Mac では、妖精さんの仕事内容が若干異なるので気をつけてください[*9]。

Ubuntu に最初から入っている「tcpdump」は妖精さんが分けたパケットや鯖（フレーム）の内容を表示したり、ファイルに複製することができます。実際に取得してみましょう。まず最初に使用しているネットワークインターフェースを ip コマンドで確認します。

```
$ ip addr
(中略)
3: wlp3s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP...
    link/ether 64:80:99:XX:XX:XX brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.15/24 brd 192.168.0.255 scope global dynamic wlp3s0
       valid_lft 82707sec preferred_lft 82707sec
    inet6 fe80::6680:99ff:fexx:xxxx/64 scope link
       valid_lft forever preferred_lft forever
```

ip コマンドの出力結果で言うところの「wpl3s0」がインターフェース名です[*10]。鎮守府の環境によっては複数のインターフェースが表示されるものと思いますが、実際に艦これサーバーへのアクセスに使うインターフェースを控えておいてください。「inet と inet6」にあるのが、割り振られた IP アドレス（IPv4 と IPv6）です。「link/ether」はそのインターフェースの MAC アドレスになります。こちらも後ほど出てきます。

インターフェースがわかったら、tcpdump コマンドでそのインターフェースを経由するパケットを記録しましょう。とりあえず艦これの「GAME START」ボタンが表示される画面を表示しておきましょう。そして次のコマンドを実行した上で、ゲームを開始します。

```
$ sudo tcpdump -i wlp3s0 -w sample1.pcap
tcpdump: listening on wlp3s0, link-type EN10MB (Ethernet), capture size 262144 bytes
```

「-i」オプションにインターフェースを、「-w」オプションに保存するファイル名を指定します[*11]。艦これを一通り

[*8] この作業を「抜猫」と呼びます。

[*9] というかそろそろ面倒くさくなってきたので、鎮守府のメタファーをやめて普通に説明することにします。

[*10] Ubuntu 15.10 からネットワークインタフェースの命名方法が変わりました。ファームウェア・BIOS の値と物理的な接続方法によってかなり異なる名前になりますので注意してください。

操作したら端末上で「Ctrl-C」を入力して tcpdump を終了しましょう。次のように取得したパケット数とドロップしたパケット数が表示されます。

```
9087 packets captured
9089 packets received by filter
0 packets dropped by kernel
```

今回は特に条件を指定せず、指定したインターフェースのすべてのパケットを取得しました。このため長時間通信するとどんどんファイルサイズが大きくなっていくので注意してください。

1.1.3 pcap ファイルから不要なデータを削除する

保存した sample1.pcap は PCAP 形式のバイナリファイルです。この中身を見るためにはやはり tcpdump コマンドを使います。

```
$ tcpdump -c 3 -r sample1.pcap
reading from file sample1.pcap, link-type EN10MB (Ethernet)
18:46:44.729953 IP 192.168.0.15.46616 > example.com.https:
 Flags [FP.], seq 2227832473:2227832600, ack 3058760346, win 362,
 options [nop,nop,TS val 880984 ecr 1753418069], length 127

18:46:44.942783 IP 192.168.0.18.ssh > 192.168.0.15.53312:
 Flags [P.], seq 3827020108:3827020500, ack 3108412220, win 1810,
 options [nop,nop,TS val 136141708 ecr 880778], length 392

18:46:44.942811 IP 192.168.0.15.53312 > 192.168.0.18.ssh:
 Flags [.], ack 392, win 1436,
 options [nop,nop,TS val 881037 ecr 136141708], length 0
```

「-c 3」で先頭の 3 パケットのみ表示し、「-r」で指定したファイルからパケット情報を読み込んでいます。出力結果は適宜改行を入れていますが、本来は 1 パケット 1 行で表示されます。角括弧（>）の前後が送信元と宛先のアドレスであり、その後にフラグ、シーケンス番号などが続き、最後のパケットのデータサイズが表示されます。

結果を見ればわかるように、HTTPS 通信と SSH 通信のみのパケットだけなので、艦これの情報はなさそうです。そこでパケットのフィルタリングを行いましょう。とりあえずポート番号が 80 のパケットのみ表示してみます。

```
$ tcpdump -c 3 -r sample1.pcap 'port 80'
reading from file sample1.pcap, link-type EN10MB (Ethernet)
18:46:54.378176 IP 192.168.0.15.60526 > 125.6.189.7.http:
 Flags [F.], seq 1477232437, ack 2440899299, win 229,
 options [nop,nop,TS val 883396 ecr 4286787085], length 0

18:46:54.378217 IP 192.168.0.15.60530 > 125.6.189.7.http:
 Flags [F.], seq 3390295697, ack 934525571, win 229,
 options [nop,nop,TS val 883396 ecr 4286787084], length 0

18:46:54.378236 IP 192.168.0.15.60528 > 125.6.189.7.http:
 Flags [F.], seq 3335369749, ack 1111981478, win 229,
```

*11 ファイルに保存せずライブでパケットの内容を表示するモードも存在します。今回は取得したパケットをあとで流用したいので、ファイルに保存しています。

Chap. 2

お得でおいしい SSL 証明書のとりかた

長南 浩 (Ubuntu Japanese Team)

2014 年 8 月に Google は Web サイト検索ランキングを決定するアルゴリズムに、「HTTPS」で情報を配信しているかどうか」を採用するという発表[*1]を行いました。

この発表の後「Web サイトの常時 SSL 化」が話題となりましたが、現状では「検索の順位を上げるためには常時 SSL」と SEO 業者が言っているわけでもない状態で 1 年半が経過しました。

パブリッククラウドや VPN を当たり前に利用できる今の時代に、どのようにサイトを HTTPS 化するのが良いかを考えてみました。

2.1 HTTPS 化ってめんどくさい?

最近はクラウドや VPS サービスなどの力を借りて、個人でも Web サービスや Web サイトを比較的簡単に作ることができるようになりました。

最近では VPS やクラウドだけでなくドメイン名も非常に安く取得できるようになったので、個人で Web サイトを持つ場合でも何か凝ったドメインでのサービス提供を行いたいものです。そして、どうせなら、そのようなサイトでも HTTPS 化したいのが人情です。

ブラウザから警告が出ない HTTPS 化するには SSL 証明書を準備する必要があるのですが、どのような証明書を買うべきなのか、できるだけ費用をかけずに入手できるのかはまだ一般的に知られていないところです。

2.2 証明書って「お高い」んでしょう?

Web サーバに使う SSL 証明書は技術的には X.509 証明書ですが、発行会社や証明書のグレードによって大きく値段が異なります。一言に「HTTPS 化の証明書」といっても Web ブラウザで信頼されるものでないと、Web サイトを閲覧したユーザには信用されません。

証明書を発行する認証局は上位の認証局への信頼を維持するためにコストをかけてセキュリティ対策を実行しいるのですが、そのコストは最終的に Web サイト運営者が発行料という形で負担するのが現状です。

また昨今では Web ブラウザに認証局と Web サイトの運営者情報が表示される EV 証明書を使うサイトも増えてきます。EV 証明書を取得するためには認証局に対して公的な書類を提出しないといけなかったり、電話でのインタビューを受けたりしないと発行してもらえません。そのような手間も証明書の値段に反映され、一般的な SSL 証明書よりも高い値段が設定されています。

それ以外にもセキュリティ上の脅威が発生した際に保証金が支払われるといった付加価値をつけている製品もありますが、その付加価値分が証明書の値段に反映され、値段の違いとなっています。

ただ、実際には同じ証明書であっても認証局によっては値段がバラバラで、結局のところは認証局のブランドにお

[*1] http://googlewebmastercentral-ja.blogspot.jp/2014/08/https-as-ranking-signal.html

金を払う部分も多い状況[*2] のようです。

2.3 一番簡単に取得できる SSL 証明書

今現在、Web サーバ用の SSL 証明書を取得しようとすると、このようなことを決める必要があります。

- SSL 証明局の選択 (証明書を作ってもらう機関の選択。ブランド力があるところはお高い)
- SSL 証明書のグレード (EV 証明書、実在確認証明書、ドメイン証明書など)
- SSL 証明書の種類 (マルチドメイン証明書、ワイルドカード証明書など)
- SSL 証明書の有効期限 (年単位で選択するケースが多い)

これらの要素を考えて「必要に応じて」取得先と製品を決めるわけですが、「できるだけ安価に」という、価格面のみ重要視で考えると、

- StartCom が提供する「The StartSSL Free (Class 1)」証明書 [*3]
- EFF と Linux Foundation が中心となる「Let's Encrypt」 [*4] の証明書
- Amazon Web Services 上のロードバランサ (ELB) と CDN(CloudFront) で無料で利用できる「AWS Certificate Manager」[*5]

がそれぞれ無料で利用することができるというサービスです。

StartSSL の Class1 証明書は個人が利用できるサービスで、ドメインの存在確認のみを行った (StartCom はグレードを「Class 1」と表現しています) 証明書で 1 年間有効なものを無料で取得することができます。

Let's Encrypt は現在「公開ベータプログラム」中ということですが、ACME というプロトコルを使って、完全自動でドメイン認証証明書を発行しています。

また、2016 年 1 月には AWS 上の ELB と CloudFront 向けに無料で SSL 証明書を発行する ACS(AWS Certificate Manager) が発表されました。ロードバランサと CDN 限定のサービスで、仮想サーバの EC2 では利用できない制限がありますが、すでに ELB や CloudFront を使っている場合は Web サービスを簡単に SSL 化することができるサービスです。

■コラム: Let's Encrypt はマルウェアベンダにも有用?

まだ「正式運用」となっているわけではない Let's Encrypt ですが、すでにマルウェアベンダ (?) も活用しているようで、2015 年 12 月に急増したとされる不正広告攻撃で、マルウェアのホスト先に Let's Encrypt の証明書が使われているというニュースが報じられている[*6] ようです。

ドメインの存在確認がされて通信が暗号化されることと、コンテンツが有害かどうかということは関係がないという至極当然のことですが、報道では「ターゲットは日本のネットバンク」とも書かれているので「ブラウザに鍵マークがついたら安全」と間違った知識が広められ、多くのユーザが信じている弱点をつこうとしているようにも思われます。最終的には「何が安全で何が危険か」を個々のユーザが判定しないといけないように思われました。

[*2] 有名な会社でブランド力があるから高い値段設定をしているところもあれば、過去にセキュリティ上の不祥事を起こしてしまいブランドイメージが悪い分値段が安い業者もあります。

[*3] https://startssl.com/

[*4] https://letsencrypt.org/

[*5] https://aws.amazon.com/jp/certificate-manager/

[*6] http://blog.trendmicro.co.jp/archives/12775

2.4 https サイトを Ubuntu で構築する

証明書を取得することはもちろん、同時に Ubuntu で SSL を利用したサービスを構築する必要があります。構築のノウハウは Japanese Team の水野さんが Ubuntu Weekly Recipe の第 387 回で「Ubuntu で SSL を利用したサービスを構築する」という記事を書かれているので、そちらを参照[*7] してください。

注意すべき点は 2048byte 以上の鍵長の秘密鍵を作成するという点と、CSR を作る際に openssl コマンドに-sha256 オプションを指定して、署名アルゴリズムに SHA-2 アルゴリズムを使うという点です。

SHA-1 署名を使った証明書は非推奨であるばかりか、ブラウザによっては「安全ではない」としてエラーや警告が表示されてしまいます[*8] 。

2.5 StartSSL で SSL 証明書を取得する

実際に StartSSL で無料で SSL 証明書を取得してみましょう。

今回は無料のドメイン認証の SSL 証明書を取得を取得するわけですが、StartsSSL ではドメインの所有者確認を、ドメインの whois 情報に記載されているメールアドレスや「webmaster@(ドメイン名)」といった、ドメイン所有者にしか使えないメールアドレスの存在確認をとることで間接的なドメイン所有確認を行っています。

SSL 証明書取得の前に、whois に記載されたメールアドレスでメールの送受信ができるように[*9] しておきましょう。

2.5.1 StartSSL アカウントの作成

StartSSL のサイトにアクセスすると、アカウントのサインアップのリンクがありますので、必要事項を記入して、アカウントを作成します。ここで指定するメールアドレスはあくまで StartSSL のアカウントのメールアドレスで、whois に指定されたメールアドレスでアカウントを作る必要はありません。

[*7] http://gihyo.jp/admin/serial/01/ubuntu-recipe/0387

[*8] SSL 証明書の期限が切れていないのに既存の HTTPS サイトがある日突然ブラウザで警告が出るようになってしまい、あわてて証明書の更新手続きを行う例が多く起こっているようです。

[*9] Value-domain やお名前.com などのサービスでは Web 上でメールアドレスの確認や変更を行うことができます。

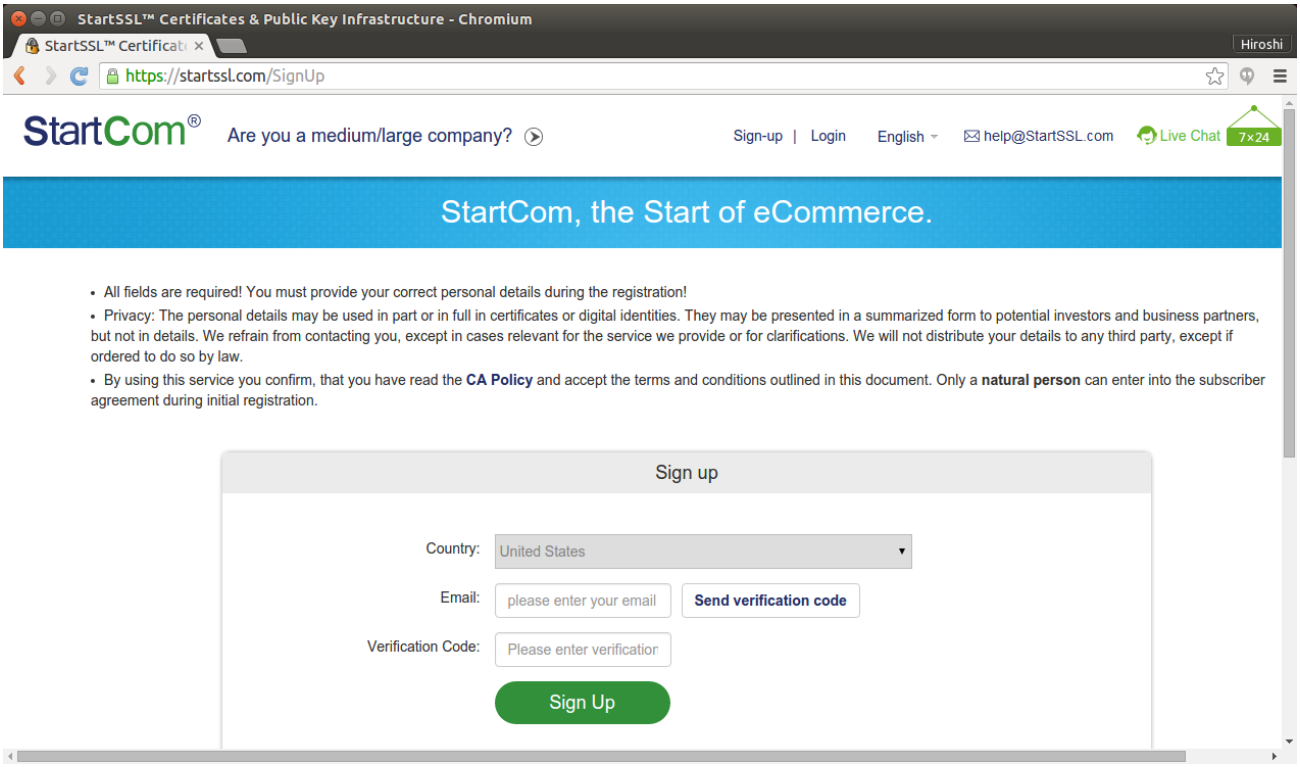

図 2.1　StartSSL のサインアップ画面

なお、StartSSL の場合はアカウントに SSL クライアント証明書を利用しており、サインアップ完了時に Web ブラウザにクライアント証明書がインストールされます。クライアント証明書は紛失しないようにバックアップをとるとよいでしょう。また、筆者が試した際には人力で記入内容を確認していたのか、時間がかかったことと、住所などを正確に書かないとアカウント作成が却下される場合があるので注意が必要です。

2.5.2　ドメイン所有の証明

次に StartSSL に対して「正当なドメイン所有者」であることを証明する必要があります。

コントロールパネル上の Validation Wizard から Domain Validation を選び、ドメイン名を入力します。すると、ドメインを取得したときに登録業者経由で whois に登録したメールアドレスや"webmaster@(ドメイン名)" などといった「ドメイン所有者が送受信できるであろうメールアドレス」が何個か表示されるので、メールを受信できるアドレスを選択し、Click to send validation code ボタンを押します。

Chap. 3

Xymon Maniax

Hajime MIZUNO(@mizuno_as)

全国 1 億 2 千万の中のほんのちょっぴりの、ざっぱーん読者のみなさんこんにちは! 最近琴似にいい地酒屋を見つけてしまいました。そこで知った、会津の遥香というお酒がとても美味しくてお勧めです。で、何の話でしたっけ?

3.1 はじめに

Xymon は BigBrother の流れを汲むネットワーク/サーバー監視システムです。実運用に必要な最低限の監視環境が、他の監視システムに比べて簡単に構築できること、テキストファイルベースの設定がシンプルで扱いやすいこと、プラグインの自由度が高いこと、外形監視がシンプルかつ高機能なことなどが特徴で、筆者のお気に入りです。Xymon についての基礎的な情報は、Ubuntu Weekly Recipe の第 383 回[*1]に執筆しましたので、まずはそちらをご参照ください。

Recipe やオフラインミーティングのプレゼンテーションでは、Xymon がどのようなツールで、どのような特徴を持っているかの解説に留めています。Xymon が持っているさらに深い機能や、運用上ハマりがちなポイント、バッドノウハウなどについては言及していません。そこで本記事では前述の情報を補完する目的で、日本語であまり深い情報がまとまっていない Xymon の運用について、筆者が数年使い続けて得た知見をまとめられればよいと考えています。

3.2 監視スクリプトを自作する

Recipe の記事を読めば基本的な監視設定はできるようになっているはずですので、インストール方法などの解説は省きます。そこでいきなり、おそらくみなさんが一番知りたいであろう、自作のクライアント監視スクリプトを利用して、独自の監視項目を追加する方法を解説します。例としてソフトウェア RAID の状態を監視し、アレイが縮退した際にアラートを出すことを考えてみましょう。

Xymon の仕組みとしては、なにかを監視して正常、異常を返すスクリプトを監視対象マシンの /usr/lib/xymon/client/ext/以下に配置するだけです。すると xymon-client がこのスクリプトを呼び出し、サーバーに結果を通知します。サーバー側ではこの監視項目に対し、アラート送信の設定をすればよいわけです。

ここではリスト 3.1 のスクリプトを/usr/lib/xymon/client/ext/raid.sh として作成しました。

リスト 3.1: RAID アレイの状態をチェックしてサーバーに送信するスクリプト

[*1] http://gihyo.jp/admin/serial/01/ubuntu-recipe/0383

```
#!/bin/sh

COLUMN=raid
COLOR=green
MSG="RAID Status"

STATUS=$(/bin/cat /proc/mdstat)
echo $STATUS | /bin/grep -e '\[.*_.*\]'

if [ $? -eq 0 ]; then
  COLOR=red
  MSG="${MSG}

RAID is degraded!

$STATUS
"
else
  MSG="${MSG}

All is OK

$STATUS
"
fi

$BB $BBDISP "status $MACHINE.$COLUMN $COLOR `date`

${MSG}
"

exit 0
```

簡単に説明すると、スクリプトは以下の三つの部分に分かれています。

1. /proc/mdstat を見て、アレイが正常であるかを判断する[*2]
2. 上記の結果によって、通知するステータスとメッセージを作成する
3. サーバーにステータスとメッセージを通知する

状態の判断とメッセージの組み立ては見ての通りなので解説の必要はないでしょう。サーバーへの通知は、変数 BB に格納されているコマンド[*3]を実行することで行われています。変数 BB は展開すると/usr/lib/xymon/client/bin/xymon になります[*4]。それではためしにこのコマンドを手で実行してみましょう。

```
$ /usr/lib/xymon/client/bin/xymon
Xymon version 4.3.7
Usage: /usr/lib/xymon/client/bin/xymon [--debug] [--merge]
 [--proxy=http://ip.of.the.proxy:port/] RECIPIENT DATA
  RECIPIENT: IP-address, hostname or URL
  DATA: Message to send, or "-" to read from stdin
```

このように xymon コマンドは引数として、Xymon サーバーの IP アドレスと、送信するメッセージを指定す

[*2] ここではブラケット内にアンダースコアがあるかないかで判断してしまっていますが、もう少し賢い方法があるかもしれません。
[*3] これらの変数名に BigBrother の面影が残っていますね。
[*4] 変数 BB は/usr/lib/xymon/client/etc/xymonclient.cfg 内で定義されています。

ることになっています。つまり変数 BBDISP は Xymon サーバーの IP アドレスである必要がありますが、これは/var/run/xymon/bbdisp-runtime.cfg 内で設定されていますので確認してみてください。

メッセージは、かならず Xymon の内部コマンドからはじめる必要があります。詳しくは xymon コマンドの man の、XYMON MESSAGE SYNTAX セクションを参照してください。ここでは単一のステータスを送信するため、status コマンドを指定しています。status コマンドの書式はリスト 3.2 の通りです。

リスト 3.2: status コマンドの書式

```
status[+LIFETIME][/group:GROUP] HOSTNAME.TESTNAME COLOR <additional text>
```

status コマンドにはまず (ホスト名).(テスト名) を渡す必要があるため、ここでは変数 MACHINE[*5]と変数 COLUMN を指定しています[*6]。COLOR は状態を表しており、"red"、"yellow"、"green"、"clear"のいずれかでなければなりません。ここではアレイが正常な時は green、縮退が発生した時に red としています。

その後には追加のテキストを任意に指定することができます。この情報は Xymon の Web 画面や、アラート発生時のメールに記載されるため、できるだけ詳しく情報を記載しておくとわかりやすくなるでしょう。ここでは/proc/mdstat の内容をそのまま載せることにしました。

これで監視スクリプトは完成です。次に、Xymon がこのスクリプトを実行できるよう、/usr/lib/xymon/client/etc/clientlaunch.cfg に、リスト 3.3 のセクションを追加します。

リスト 3.3: clientlaunch.cfg に追記する内容

```
[raid]
        ENVFILE $XYMONCLIENTHOME/etc/xymonclient.cfg
        CMD $XYMONCLIENTHOME/ext/raid.sh
        LOGFILE $XYMONCLIENTHOME/logs/raid.log
        INTERVAL 5m
```

ENVFILE は、このスクリプトを実行する際に読み込まれる設定です。前述の通り BB や BBDISP といった変数を利用するため、$XYMONCLIENTHOME/etc/xymonclient.cfg を指定してください[*7]。CMD は実行する監視スクリプトです。先ほど作成した raid.sh を指定してください。LOGFILE は説明するまでもないでしょう。なお$XYMONCLIENTHOME/logs はシンボリックリンクになっており、実体は/var/log/xymon を指しています。INTERVAL は監視を行うインターバルです。

これで終了です。Xymon クライアントが CPU や DISK といった情報と一緒に、RAID の情報も Xymon サーバーに通知してくれます。便利なのは、サーバー側では「RAID を監視する」という設定を明示的に行う必要はないという点です。たとえば HTTP や SSH の外形監視などは、サーバーがクライアントに対して疎通を確認しに行くため、サーバーの hosts.cfg に監視項目をすべて列挙する必要がありました。しかしクライアント監視スクリプトによって実行される監視項目は、クライアントが自発的にすべての項目を通知してくるため、監視項目をサーバーで指定する必要はありません[*8]。

[*5] 環境変数 MACHINE は BB や BBDISP とは異なり、/etc/init.d 以下の起動スクリプトによって設定されています。Xymon の Upstream では uname -n | sed -e 's/\./,/g' の結果を使いますが、Debian/Ubuntu のパッケージでは/etc/default/xymon-client で指定されている変数 CLIENTHOSTNAME をそのまま使うようになっています。

[*6] 変数 COLUMN はこのスクリプト内で指定している監視項目名です。ここで指定した名前が、新しいカラムとしてサーバーの監視画面に生えてきます。任意の名前を指定してかまいません。

[*7] 変数 XYMONCLIENTHOME 自体も、xymonclient.cfg 内で設定されています。

[*8] 監視項目をなにも指定しなくても、cpu や disk、memory の情報が通知されていますよね。それと同じです。

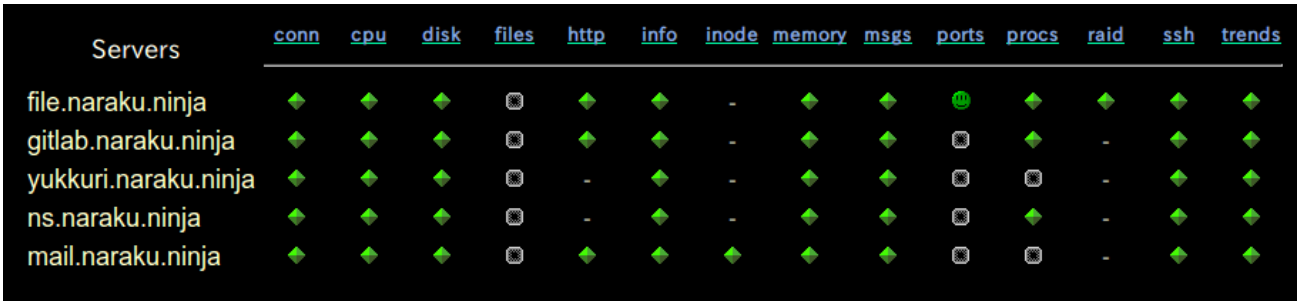

図 3.1　クライアント監視スクリプトを足した後の状態。RAID カラムが増え、ファイルサーバーの RAID の状態が監視されているのがわかる。

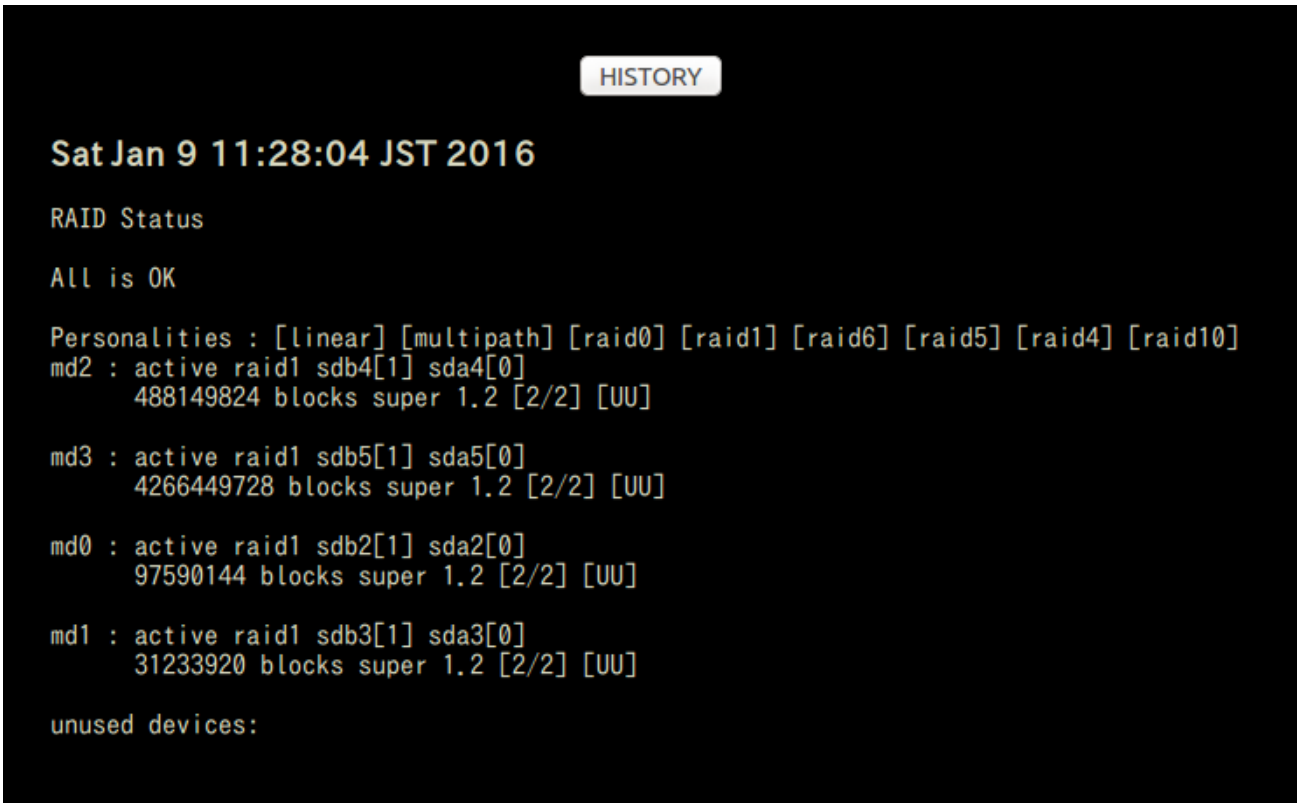

図 3.2　詳細ステータス画面。スクリプト内で設定したメッセージが表示されている。

3.3　プロセスやネットワークを監視する

Xymon クライアントがインストールされたマシンを hosts.cfg に追加すると、cpu、disk、memory といった項目が自動的に監視されるのは何度も述べてきました。しかしその中にある files、ports、procs といったカラムが、白色 (clear) になっているのを疑問に思った方もいるのではないでしょうか。これらは特定のファイルやポートの状態、プロセスの存在などを監視する機能ですが、デフォルトでは具体的な監視対象が設定されていないため、白色になっているのです。

analysis.cfg は、Xymon クライアントから送られてきたレポートに対して、しきい値や追加の監視項目、無視する項目などをホストごとに設定するファイルです。たとえばファイル末尾にはリスト 3.4 のような記述があります。

リスト 3.4: 全サーバーに対して適用される監視項目としきい値

Chap. 4

Ubuntu と知の巨象、Evernote との共存をめぐって

Rakugou

Evernote[*1] といえば、言わずと知れたアプリケーションです。テキスト、画像、音声、ドキュメントファイル、ウェブサイト、などなど、何でも保存して、そこから必要な情報を検索によって探しだせる、知の貯蔵庫。もはや第二の脳と言っても過言ではないツールです。何百万人と利用している大人気アプリケーションを、公式アプリケーションが現時点では存在しない Ubuntu で使用する方法を模索していきます。

4.1 Evernote 氏〜Ubuntu でも使える公式アプリケーション出してくだされ〜頼む〜

Ubuntu を愛用されている皆様におかれましては、スマートフォンで Evernote を見られていることも多いと思われます。しかし、パソコンの大画面で大きく、多くのノートを見たい、あるいは、スマートフォンによるフリック入力より、タイピングした方が入力スピードが早いので、パソコンでノートを作成したいとなると、どうしても Ubuntu 版のクライアントが欲しくなるというものです。しかし、現段階において、Ubuntu 向けの Evernote 公式アプリケーションが存在しておりません。これは辛いです。筆者のように Ubuntu で安心安全に Evernote を楽しみたいファンといたしましては、ぜひとも公式アプリケーションがほしいところです[*2] [*3]。

4.2 非公式アプリケーション

公式アプリケーションは存在しないものの、非公式アプリケーションはいくつか存在します。その中から代表的なものを紹介していきます。

4.2.1 Everpad

かつて、Ubuntu Magagine Japan Vol.10 にて紹介されていた非公式アプリケーションでもあります。筆者もこのアプリケーションが使いやすくて一時期愛用していました。PPA で配布されていましたが、15.04 を最後にそれ以降のバージョンについては Launchpad で公開されておりません (14.04 では使用できることを確認済みです。また、12.04 についても、Launchpad に情報があることは確認しております。)。14.04、12.04 の LTS を使用している場合は

[*1] https://evernote.com/intl/jp/

[*2] Evernote の見解によると、「Evernote は、現在 Linux には対応しておりません」とのことです (「現在」というフレーズが重要です)。いずれは、いつの日か、Linux 対応のクライアントが導入されるかと信じています。

[*3] ダウンロードページ（https://evernote.com/intl/jp/download/）を Ubuntu から見ていただけるとわかるかと思います。

```
$ sudo add-apt-repository ppa:nvbn-rm/ppa
$ sudo apt-get update
$ sudo apt-get install everpad
```

上記のコマンドを用いてインストールしてください。画面自体は至ってシンプルです。無駄のない構成になっているのがわかるかと思います。

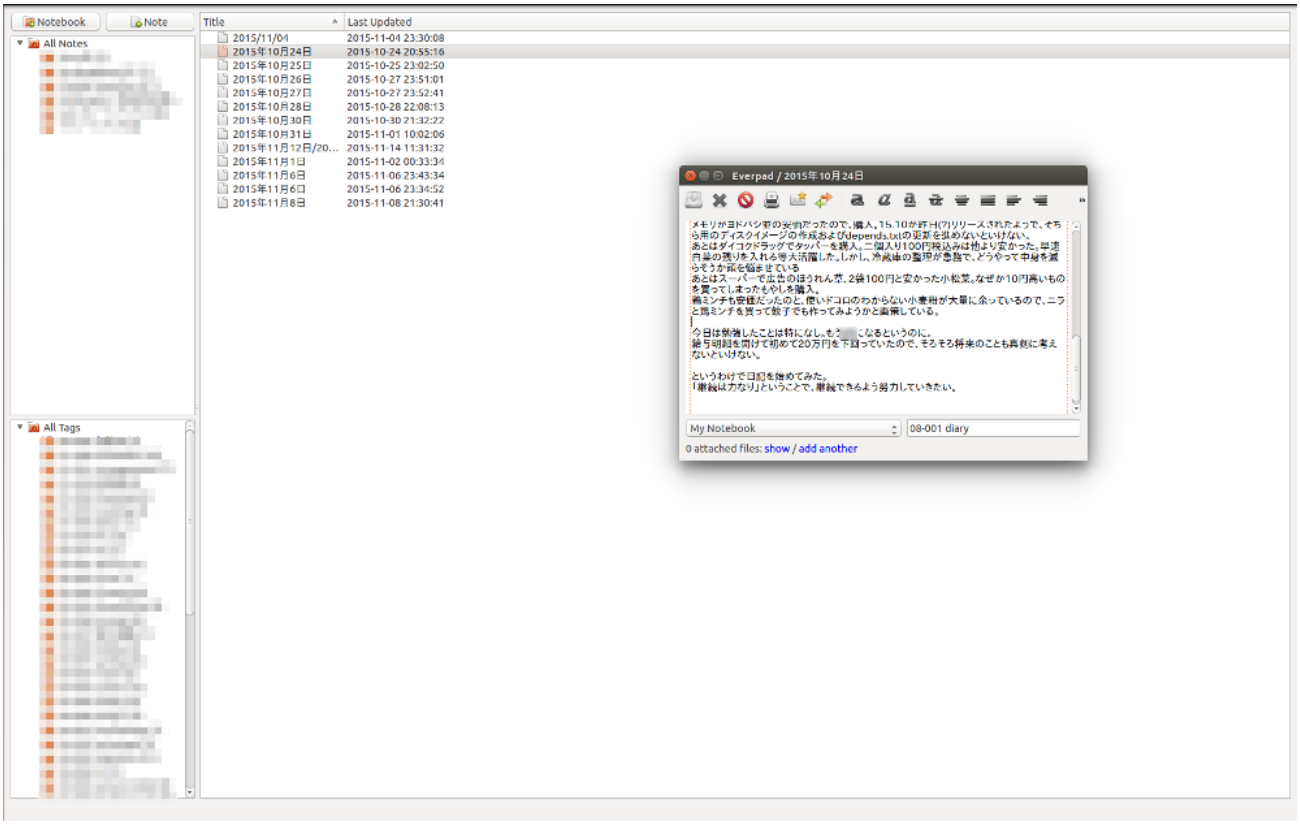

図: Everpad の画面、画面の余白からいかにシンプルかというのがわかるでしょう。

今回のプレビュー作成にあたり、VirtualBox にインストールしている 14.04 から起動してみたのですが、もたつくものの思った以上にあっさり起動できたほどでした。メイン画面から「Create Note」ボタンを押すことによってノートを作成することができ、考えたことを即座にノートに移すことができます。ボタンがそれほど多くないのであまり迷うことがないというのが嬉しいところです。また、ノート作成画面で、タグ付け、ノートブックの整理、チェックボックス、画像挿入、箇条書きなどといった基本的な設定はできるようになっています。細かい設定をあまりいじることのない筆者にとっては、使いやすさは群を抜いていますが、15.10 で使えないというのは難点ですね[*4]。

4.2.2 Neighbornote

Neighbornote[*5]は Nixnote をベースに開発された、連想機能を持つ非公式 Evernote アプリケーションです。

事前に Java と OpenSSL の準備が必要で、下記コマンドを入力、あるいはソフトウェアセンターで検索し、インストールします。

[*4] 特に筆者が愛用している Let's note では 14.04 ではインストールすると不具合が生じる（ざっぱーん Vol.3 参照）という難点があるため、15.10 では使えないというのは深刻な問題なのです。

[*5] https://neighbornote.herokuapp.com/index.html#home

```
$ sudo apt-get install libssl-dev openjdk-7-jre
```

そして、ダウンロードページ*6からインストールします。64bit 版はファイル名に x64 がついています。インストールした後、zip ファイルを解凍し下記コマンドを入力します。（ダウンロードし、解凍されたファイルがダウンロードフォルダにあると仮定します。）

```
$ cd ~/ダウンロード
$ chmod +rw neighbornote-0.5.4-linux-x64-installer.run
$ ./neighbornote-0.5.4-linux-x64-installer.run
```

この後、インストール画面が出てくるので、指示にしたがってインストールします。インストール後、デスクトップにショートカットが作成されておりますので、アイコンをダブルクリックすると、Neighbornote が起動します。画面が下記の通りとなっておりますが、非常に見やすいです。

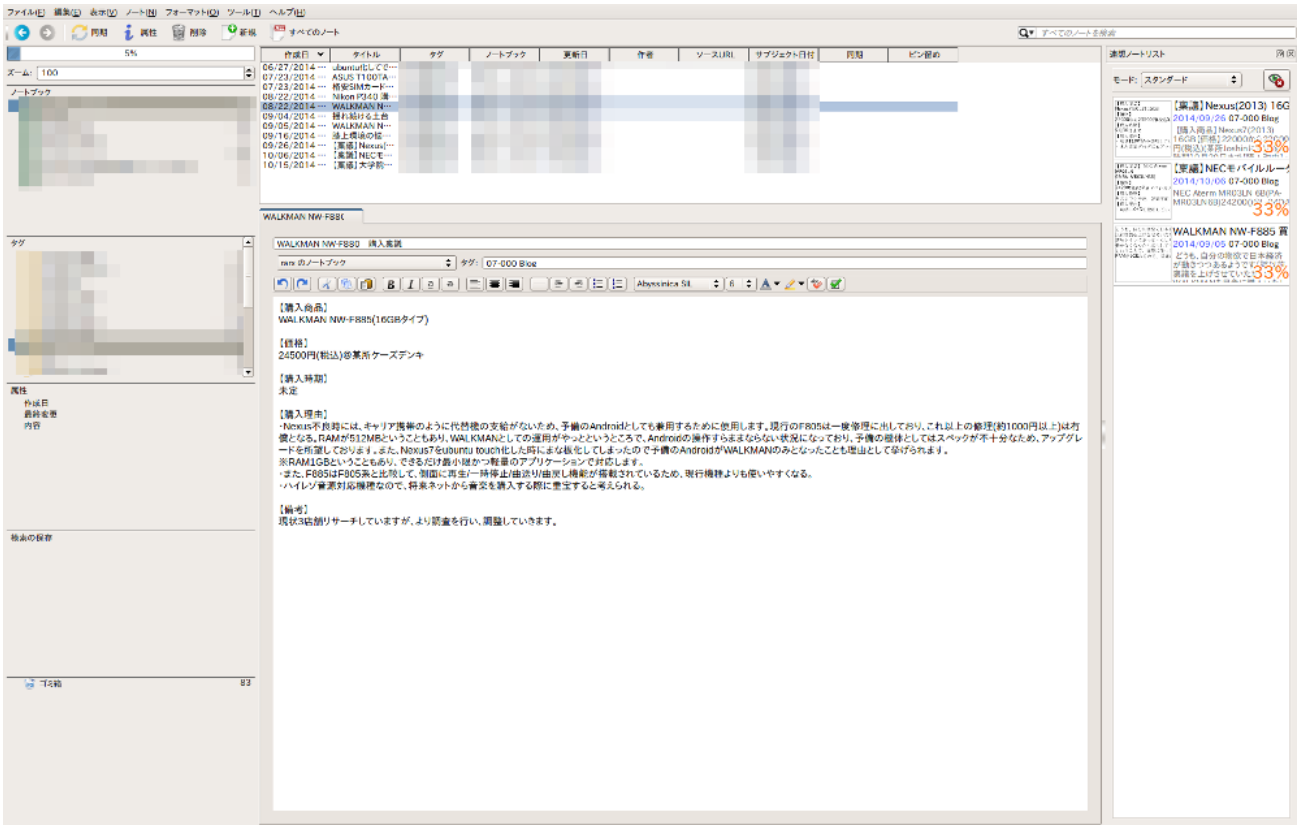

図: neighbornote の画面、ルックス的には Nixnote に近い印象を受ける。

また、画面右側に見えるのは「連想ノートリスト」で、見ている記事に類似したノートをリストにしてくれる機能です。ノートの内容や、過去にどのようにノートを編集、整理していたかの近似性を元に、ノートを抽出して表示してくれます。暇なときに連想ノートを数珠つなぎに自分のノートを見てみるのも、知識のネットワークをたどることで、「故きを温ね、新しきを知る」と言ったように、新しいアイデアが生まれていいかもしれませんね。連想ノートの連想のさせ方は「モード」タブを切り替えると変更することができます。ちなみに、「カスタム」を選択すると、自分で頻

*6 https://neighbornote.herokuapp.com/download.html

度を設定することができます。「編集」から「設定」をクリックし、設定画面左側にあるアイコンの「連想ノートリスト」をクリックすると、連想設定を変更することができます。

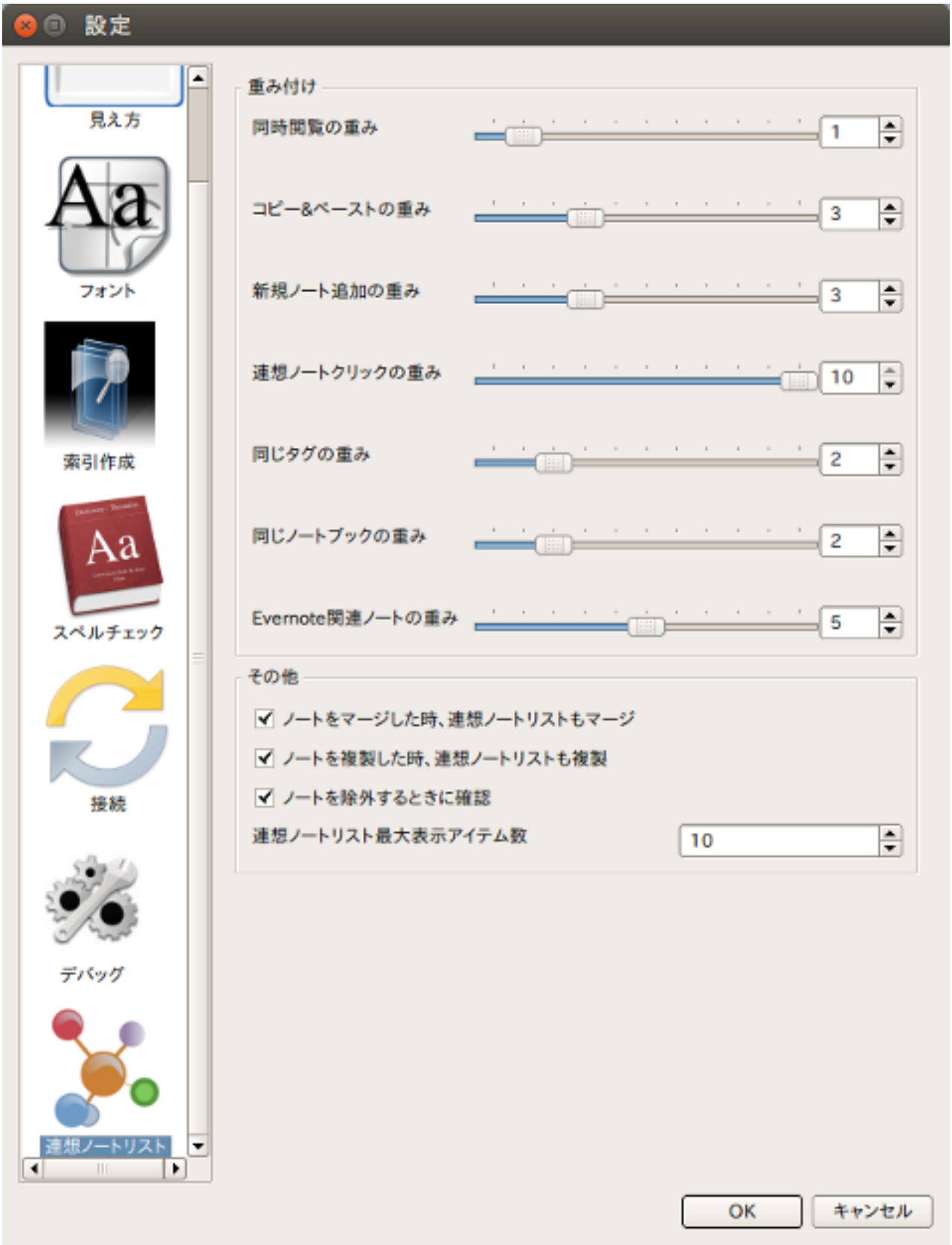

図: 連想設定を変更する画面。重み付けを変更することで、連想リストに出てくるノートが違ってくる。

Chap. 5

ゲストページ

5.1 Ubuntu GNOME の再インストールと棚卸し

あわしろいくや

今回のテーマであるところの「ネットワーク」を決めたのは、言うまでもなく筆者なわけです。が、それを筆者自らぶっちぎるのは、いつものことなのか、それとも笑うところなのか (vol.3 からのコピペ)。

5.1.1 理由

昨年 12 月上旬にメインで使用している PC の HDD が故障しました。OS は Ubuntu GNOME 14.04 LTS でした。急遽同じ型番の HDD を購入し、dd でセクタレベルでのコピーをして使用するつもりでしたが、うまくいかなかったので再インストールし、必要なファイルをコピーすることにしました。

というわけで、期せずして環境を再構築することになったのですが、具体的にどんなパッケージをインストールし、どんな設定をしたのか棚卸しすることになりましたので、今回はその様子をお届けします。

5.1.2 インストール

インストールする OS を Ubuntu GNOME 14.04 にするか、15.10 にするかは迷いました。しかし、Ubuntu Weekly Recipe 第 396 回　Ubuntu GNOME 15.10 の変更点で 15.10 がとてもよさそうであることを知ったので、こちらにしました。

iso イメージをダウンロードし、これを USB メモリーに転送する必要があるのですが、当時は Ubuntu のリポジトリにある usb-creator が壊れていて転送したイメージがブートできないため、Windows に Universal USB Installer をダウンロードし、こちらを使用しました[*1]。

故障した HDD は何度となく dd でコピーする方法で変更してきており[*2]、最初にインストールした PC は UEFI 非対応だったのでずっとそのまま来ていました。今回は新規インストールということで、UEFI でブートするようにしましたが特に意味はありません。

インストールが完了したら、/home/ikuya 以下と/var/vm 以下を rsync でコピーします。自分でも理由は覚えていませんが、VirualBox のイメージは/var/vm に置くようにしています。幸いにも/home/ikuya 以下は全てのファイルが無事でした。/var/vm 以下はコピーできないものもありましたが、その場合は仮想マシン自体を削除するか別のバックアップから書き戻すかしています。

[*1] あとからよくよく説明を読むと Wine でも動作するらしいですが、USB メモリーを認識できるんでしょうか。

[*2] 古い HDD と新しい HDD を接続し、古い HDD から新しい HDD に dd でコピーします。当然古い HDD より新しい HDD のほうが容量が大きいため、コピーしただけでは新しい HDD の全ての領域が使用できるわけではありません。よって GParted を使用して領域を拡大する、という方法で数台の HDD を乗り換えてきました。

5.1.3 追加パッケージ

リポジトリにあるパッケージでインストールしたのは以下のとおりです。

```
vim vlc ssh lv chromium-browser shotwell byobu git vinagre gimp virtualbox
virtualbox-guest-additions-iso owncloud-client clipit bijiben asunder
light-themes easytag gedit-plugins epiphany-browser fonts-ipafont subversion
imagemagick samba retext
```

Mozc はデフォルトでインストールされていますけど、自分でビルドした[*3]より新しいバージョンにアップデートしています。

リポジトリにないパッケージとしては、Atom と Pandoc をインストールしています。正確にいえば Pandoc のパッケージはリポジトリにありますけど、更新が多いわりに自力でのビルドが大変なので、パッケージをダウンロードしてきてインストールするのが簡単だと思います。実のところ Atom はあまり使わないのですが、それでもインストールしています。

SoftEther-VPN もないと生きていけないため、インストールしています。しかし現在公開されているソースコードは systemd には対応していないため、自力で対応しました。github.com/ikunya/SoftEtherVPN にあります。

5.1.4 /etc/

/etc/は当然のことながらデフォルトに戻ってしまったので、復旧する必要があります。幸いにも故障した HDD にある/etc/も無事だったため、フォルダーごと tar に固めて新しい HDD にコピーし、必要なファイルだけコピーすることにしました。samba と ssh は定番ですが、それ以外には/etc/profile.d/unzip-default-charset.sh と/etc/udev/rules.d/99-udisks2.rules もコピーしています。前者は unzip でファイル名の文字コードをよしなにしてくれるもので、後者は/media/(ユーザー名)/(UUID またはラベル名) となるものを、/media/(UUID またはラベル名) に変更するものです。前者は Ubuntu Japanese Team の PPA で公開している unzip パッケージに収録されていますが、後者はどんなものかわかりづらいと思いますので、ここに掲載します。

```
$ cat /etc/udev/rules.d/99-udisks2.rules
ENV{ID_FS_USAGE\}=="filesystem", ENV{UDISKS_FILESYSTEM_SHARED\}="1"
```

5.1.5 ルック＆フィールの変更

インストールした GNOME Shell の拡張機能は、"Dash to Dock"と"topicons"の 2 つのみです。Weather は使うのをやめて単独アプリの"天気 (gnome-weather)"を使用するようにしたのですが、あまりに不便なので拡張機能に戻るかも知れません。

Tweak Tool を起動し、[外観] タブの [GTK+] を [Ambiance] に、[アイコン] を [Human-Dark] に変更しました。ちなみに Ambiance テーマは light-themes パッケージにあります。もともと Ubuntu だったのを Ubuntu GNOME にしたせいか、テーマは Ubuntu っぽいのが好みのようです。

何故かタイトルバーのアイコンが左から動かなかったため、次のコマンドを実行して最小化・最大化・終了のボタ

[*3] 自前でビルドした Mozc のパッケージは公開していません。ビルド方法は vol.3 参照ですが、古くなっちゃったのでなんとかしないとですね……。

記録に残るものが記憶に残るもの　～あとがきに代えて～

あわしろいくや

私は博物館が好きで結構いろんなところに行ってます。最近だとリニューアルした江戸東京博物館、横浜開港資料館、ちょっと前だと宇治市源氏物語ミュージアム、大津市歴史博物館なんてところにも行ってます。東京国立博物館は大好きで、そろそろ 2 桁台に乗りそうなくらい通ってます。もし子供の頃に行っていたら、おそらく人生観が変わっていたでしょう。

そこで思うのは、記録に残るものが記憶に残るものであるということです。江戸東京博物館は写真撮影できますが、横浜開港資料館はできませんでした。というわけで、江戸東京博物館の展示物はどんなものがあったか、またはどんなものに興味を引いたかというのは手元に写真があるから思い出せるわけですが、横浜開港資料館は写真撮影できなかったので、展示物のことは時とともに忘れていきます。それでも図録があればまだいいですけど、ない場合は本当にどうしようもありません。大津市歴史博物館、君のことだ。特別展は図録に残りやすいので、写真撮影不可でもやむを得ないかなと思わなくはないですが。

展示物保護の観点から撮影できないというのは、話としてはわからないでもないのです。でも、フラッシュを使用しなければいいだけの話です。おそらく日本一の博物館であろう東京国立博物館は写真撮影可能であることから、この説の説得力が弱いのは事実です。何度も来てほしいからということもあるのかも知れませんが、記録に残せないところに何度も行く気にはなりませんし、そもそも実物が見たいから博物館に行くのであり、そこいらの写真でいいのなら博物館に行く必要がありません。あとは権利者への配慮とかでしょうか。博物館に置くようなもので権利者にどんな配慮が必要なんでしょうか。江戸東京博物館なんて Windows 95 のパッケージを展示してありますよ！

記録に残るものが記憶に残るもの、という話の端的な例は源氏物語でしょう。源氏物語の原本は現存していませんし、平安時代にコピーなんてできなかったわけで、複製するためには一冊一冊写本するしかなかったわけです。面白いのは、ただ写すだけじゃなくて自分なりのアレンジを加えていて、それが各所にあるものだから、それらを総合してもともとのストーリーはこんなだったに違いないと推測したものが現在知られている源氏物語というところです。まさに記録に残っているからこそ記憶に残っているわけですね。

もちろん、記録に残しているけれど応仁の乱で燃えてしまった、関東大震災、太平洋戦争、阪神・淡路大震災、東日本大震災その他天変地異等々によって消失したということも少なくないですし、記憶に残るだけで記録に残っていないエピソードもたくさんあるわけではあります。大阪市街が空襲に遭っていた際、比較的被害が少ない梅田などに市民を逃がすため、深夜にもかかわらず地下鉄を走らせたエピソードが記憶に新しいところですが、記録としてまとめられたのは戦後 50 年以上経ってからのことだったとのことです。

私もこの世界（どの世界？）に長くいるもので、いろいろと記録に残したいことがあります。私自身が関わったことであれば自分で記録すればいいだけで、それは機会を見つけて今までもやってきましたし、今後もやっていくつもりです。

ただ、記憶というものは忘却を含めて日々変わっていくものです。あの時はしんどかったけど、今思えばいい思い出だ、なんてのもよくある話です。もちろん今思い出しても腸が煮えくり返るわってこともありますけど。そして、立場が変われば見方も変わるものです。よって複数人の証言を取るのが精度を高める方法であるのは言うまでもないでしょう。

人を動かすことはお金がかかることです。皆霞を食って生きているわけではないので当然のことなのですが。博物館も無料のところはありますけど、たいていは入館料を取られますし、場合によってはそれでも足りずに税金が投入されることもあります。そこまでしてでも維持されるべき公共物を保有しているのは、それだけ文化レベルが高いということであり、ステータスであるわけです。

一方、我らが『ざっぱ～ん♪』は、残念ながら売上が芳しくなく、多額の原稿料をお支払いして広く原稿を集められる状況にありません。それでも少しずつお金をため、理想に近づいていきたいと考えていますし、みなさまのお力添えを頂戴したいところであります。

世の中理想どおりに行かないなんて普通のことですし、それでもそこに近づくために一歩々々前進し、今というスナップショットを原稿という記憶に落とし込むために日々書き続け、さらにそのスナップショットとしてこの『ざっぱ～ん♪』をお届けし、さまざまな記録を残していきたいと考えております。

最後までお読み頂きましてありがとうございました。次号は Ubuntu 16.04 LTS がテーマです。またお会いしましょう。

うぶんちゅ！　まがじん ざっぱ〜ん♪ vol.4 体験版

著　者　　うぶんちゅ！　まがじん ざっぱ〜ん♪ の仲間たち
イラスト　よかぜ
発行所　　うぶんちゅ！　まがじん ざっぱ〜ん♪ 編集部