ささみのほんのみほん

見本のため、各2pずつとなります。
PDF本編では、一部の画像が
カラー画像となります。

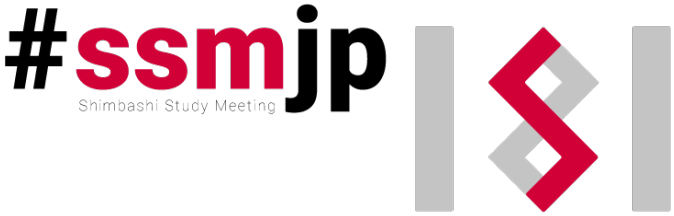
Archives

ささみのほん Archives

ssmjp 同人部　制作

2023-05-20　　発行

ささみのほん Archives のご紹介



ささみのほん Archives を手に取っていただきありがとうございます。
ssmjp 運営のじゅんじゅんです。
この本は

- 2017 年の技術書典 2 で頒布したささみのほん
- 2018 年の技術書典 5 で頒布したささみのほん 2
- 2019 年の技術書典 6 で頒布したささみのほん 3
- 2019 年の技術書典 7 で頒布したささみのほん 4

をまとめた本になります。Archives という形にするにあたり、一部手直しや加筆修正が行われています。当時買っていただいた方は見比べてみるのも楽しいかもしれません。

ssmjp とは

大体 2009 年 6 月から月一回開催される謎の街角ゆるふわ勉強会で、テーマは運用となっていますが面白ければ IT に限らず様々なテーマでカオスな LT が展開されます。
(カオスな一例)

- 養鶏の話
- 賃貸の水回りの話
- さっき初めて使ったツールの話 etc…

もちろん

まじめなお話もあります。
(まじめな一例)

- PCIDSS の話
- 運用ツールの話
- 物理を含むセキュリティの話

また、他の勉強会との合同開催なども行っております。
興味があるかたは是非遊びに来てください。

勉強会公式サイトはこちらの QR コードから

執筆者募集のおしらせ

ssmjp 同人部では、今後も執筆者を募集します。
まずは ssmjp にご参加いただいて、場のノリを体感していただき、そのあとで記事をかいてみませんか？
毎月開催の ssmjp については https://ssm.pkan.org/ にて告知いたします。
同人部としては Slack にてやりとりを行っておりますので、Slack にもご参加下さい。

お問い合わせ

この本に関するお問い合わせは、こちらまでお願いします。

- URL: https://ssmd.pkan.org/
- Mail: ssmd@pkan.org
- Twitter: @ssmjp_doujin
- misskey: @ssmjp_doujin@misskey.haun.jp

目次

第 1 話

意識低い勉強会運営の話:改 2

かいたひと：じゅんじゅん (ssmjp 受付、広報担当)
Mastodon: @junjun@mstdn.haun.jp
@junjun@janogdon.net
Twitter: @junjun_

ゆるい運用系勉強会の #ssmjp の受付と広報を担当している私「じゅんじゅん」が昨年の Advent calendar に書いた記事をもとに、今回もうちょっと詳しく説明してみようと思います。
#ssmjp というのは毎月 1 回、どこかに会場を借りて運用にまつわるテーマで話をする勉強会です。
2009 年 6 月に始まり、現在まで毎月開催されています。

勉強会の運営ってどんなことするの?

> 運営- 団体などの機能を発揮させることができるように、組織をまとめて動かしていくこと。
> “運用”, デジタル大辞泉,goo 国語辞書, 入手先<http://dictionary.goo.ne.jp/jn/>

とあるくらいなので勉強会を開きたいと思った人 (達) が計画的に準備して開催することです。
ゆるふわ勉強会の#ssmjp の運営について事前に準備すること、当日までに行うこと、開催後に行うことについてまとめてみました。

事前準備

登壇者確定

最近の#ssmjp は喋りたい！ という人からの挙手制としており、それを 2 時間で埋まるように調整しています。
2 時間きっちりでスケジュールを組むと飽和するので、多少短く見積もるのがコツです。
登壇者自身が予定通りに喋りきれず時間が溢れたり、質疑応答が発生したり、機材トラブルで予定通りに始まらないことで予定していた時間を超過してしまいます。
また、#ssmjp は運営スタッフが 4 人おり各自が喋りたいと挙手を受けるので Google スプレッドシートにて誰が何分喋るかを記載し、管理しています。

日程調整/会場探し

登壇者の都合と運営陣の都合が合う日を調整をします。登壇者との日程調整にはスケジュール調整サイトを利用し複数日を提示して全員の都合が合う日を決定します。
#ssmjp では 調整さん (https://chouseisan.com/) を利用することが多いです。
また、当日までの連絡ツールとして Slack や Facebookメッセンジャーを利用します。
会場は運営陣の伝手を使い、開催日にお借りできるところを押さえます。

当日まで

情報公開

#ssmjp では Wordpressと connpassで情報を公開しています。
#ssmjp が始まった当初はメールのみでしたが、後に Redmineに移り今の Wordpressに落ち着きました。
なお、#ssmjp は毎月開催なので当月の開催中に次回の告知と同時に情報公開をして、次に参加される方の初動申込数を稼ぐ作戦を取っています。
connpassを使うことで受付番号を発行でき、当日の受付処理の負担を軽減しています。

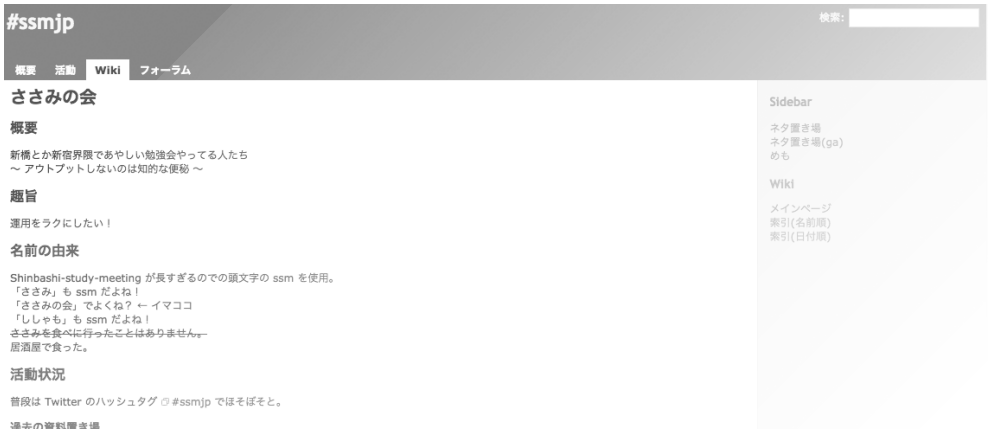


図 1.1: Redmine時代の#ssmjp 情報公開サイト

続きは本編にて！

第 2 話

Mastodon を運用してみてる話

書いた人
じゅんじゅん (#ssmjp 受付、広報担当)
Twitter:@junjun_
Mastodon:@junjun@mstdn.haun.jp , @junjun@janogdon.net

ささみのほん 2 おめでとうございます！

前回につづいて、立候補したもののテーマがいつまでも決まらなかった受付のじゅんじゅんです。今回は、1 年半くらい Mastodon のインスタンスを運用している話を書こうと思います。

何故 Mastodon インスタンスを立てようと思ったか

身内の irc で Masotodon の話題が盛り上がり始めた 2017 年 4 月上旬、さくらのクラウドお試し券を持て余していた私はこのお試し券で Masotodn のインスタンスを立ててみようと思い立ちました。ちょうど mstdn.jp でサーバが個人宅からさくらインターネットへ切り替わった直後のことです。

構築で躓きました

当日夕方に、docker が動くところまではできていました。

図 2.1: 当日のチャットのログ

irc の参加メンバーで楽しく作れば、楽しい運用ができるかなと思いみんなにアカウントを付与しあれこれといじりながら作業をしていきました。初日は結局上手く動かなかったけど、翌日には何とか形になりました。

図 2.2: 翌日のチャットのログ

続きは本編にて！

第 3 話

おうちインフラ運用事情の話

かいたひと：じゅんじゅん (ssmjp 受付、広報担当)
Mastodon: @junjun@mstdn.haun.jp
@junjun@janogdon.net
Twitter: @junjun_

ささみのほん 3 おめでとうございます!(出る前提)。

ssmjp の広報っぽいこととか受付とか会場調整とかをしているじゅんじゅんです。3 を出そうとなった際に、手を挙げたもののやはり書くことに迷っていたところ

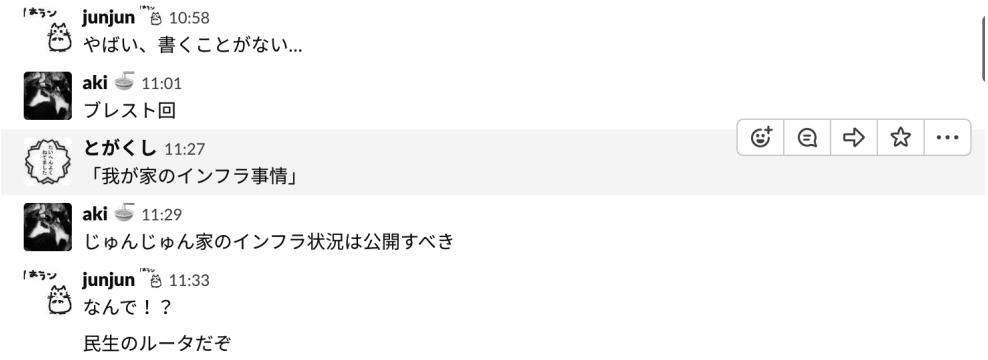


図 3.1

といわれたので、そんな話を書いてみようと思います。なお、民生機ばかりなのであまり面白いことはありません。

おうちのインフラ事情 ネットワーク編

民生機です。BUFFALO の WZR-1750DHP2 を使っています。主に買い換えるときの理由が壊れたから買い換えるなので、ほんとはもうちょっと機器を厳選すべきなのでしょうが、今のところ壊れていないのでそのまま使っています。NAT ループバックに対応してるのもまたいろいろと手を抜けてありがたいです。来客用に SSID を分けられるというのも嬉しいんですが、今のところその機能を使ったことはありません。家庭内は/24 で第 4 オクテットの

- 2-99 が固定エリア

- 100-220が DHCP で困らない機器
- 220-255がまた固定エリア

にしています。当初は、最初の固定エリアを物理マシン、最後の固定エリアを仮想マシンで使用する予定でしたがなぜか録画サーバが固定エリアに存在していていきなり破綻しています。外部からは DMZ である pkan.orgのサーバへ向けています。民生機でも syslogを転送する機能があるので、ログは DMZ のサーバへ送るようにしていますが、今のところこのログを見て何かをするような事はありません。

おうちのインフラ事情 サーバ編 1

いわゆるご家庭のパソコンに Linux を入れて使っています。ssmjp や同人部の Wordpressが動いているのは先ほどのネットワーク編で DMZ に転送していると書いたサーバになります。今日日クラウドで運用するのが一般的ではありますが、家にログインできたほうが都合が良い面もあるためおうちでのサーバ運用は止められないなと思っています。(とはいえ、公開コンテンツは外の方が良いんだろうなあ…)

おうちのインフラ環境 サーバ編 2

おうちで常時稼働してる Linux 機は、物理が 4 台と仮想が 4 台。VPS で 3 台仮想を借りているのにそんなにいるのかといわれると不要な気がしますが…物理機としては、先ほど書いた DMZ にいるおうちのゲイトウェイとなるサーバの他に、録画機とファイルサーバとツレのおもちゃです。仮想マシンは、ファイルサーバ上で動いています。DMZ では CentOS7.6 (2019/3/23 現在) で、httpd とメールサーバが動いておりささみのほん 2 で登場した Gitlab や zabbix はこのサーバで動いています。ささみのほん 2 で登場した Redmineはこのサーバではなく Apacheのリバースプロキシを使い仮想へ転送しています。この DMZ にあるサーバが死んでしまうと、同人部や Gitlab も zabbix もそうですが Redmineにまで影響がでることを考えると機器構成を見直した方が良いかもしれません。(見直すとは書きません)

おうちのインフラ環境 サーバ編 3

Linux 機はと書きましたが、Windowsサーバ機もあります。もっとも、Windはライセンスが高く簡単に手が出せるものではないことと、あくまでも評いことから評価版のライセンスを使っています。180日という試用期間きる (つまり 2 年は使える) ので、2 年間評価でのみ使うのであれば手に解釈しています。Hyper − V も試用で使えるので母体 (Wンを 2 台 (Windows2008R2/Windows2012R2)建てて Win(覚えられるとは書いていない) 思っています。

続きは本編にて！

第 4 話

勉強会を運用する話

かいたひと：じゅんじゅん (ssmjp 受付、広報担当)
Mastodon: @junjun@mstdn.haun.jp
@junjun@janogdon.net
Twitter: @junjun_

ささみのほん 4 おめでとうございます！ (出る前提)。

最近、とあるイベントの運営としてお手伝いをしてきたのでその辺の話を書いてみようと思います。

ssmjp の運営については、以前「意識低い勉強会運営の話:改 2」に書きましたのでこちらを参照して下さい。ささみのほん Archives に掲載されています。(と既刊の宣伝も忘れない)

何のイベントの運営を手伝ったのか。

Vuls 祭りというイベントのお手伝いをしました。Vuls 祭りとは、Vuls というセキュリティスキャナのユーザミートアップ的な位置づけです。昨年夏に第 4 回、今年の 6 月に第 5 回を開催しました。

私個人は、第 3 回に参加者として、第 4 回と第 5 回はスタッフとしての参加です。

なんでスタッフになろうと思ったか。

以前 Vuls を家に導入しようとしてくじけたため、Vuls 実は使ったことありません。会社でも仕事の環境で FutureVuls を紹介して使ってみましょうという流れにもなりましたが大人の事情で頓挫しております。そんな背景もあり、OSS への貢献もできてないのでイベントでスタッフとして貢献できればなあと思った次第です。

普段から Vuls の Slack に常駐していたので Vuls 祭りやるんだよという話が出たときにスタッフに立候補しました。元々、Vuls 開発者の神戸さんとは ssmjp を通して面識があり、ssmjp での受付の実績を認めてもらいスムーズにスタッフとして参加できました。

Vuls 祭りでの自分の役割

Vuls 祭り#4 では受付として、Vuls 祭り#5 では引き続き受付と、私の所属している会社に頼んでドリンクスポンサーを担当しました。

受付担当がやることは ssmjp でも Vuls 祭りでもさほど変わりません。connpass で発行する受付番号と照合して出欠確認のチェックを入れていく。また、Vuls 祭りは有償の

イベントなので受付で代金を徴収することと徴収した証拠のシールを配布します。

ssmjp ではあまり発生しない金銭の受け渡しなどがある分オペレーションは増えますが、オペレーションが混乱するのは受付列が多いときくらいでした。(コレは問題点ですね)

Vuls 祭りではスポンサーを募集しており、スポンサーとして LT をすることができます。本来は Vuls を使う話をするところではありますが、先にも書いたとおり弊社でも今のところ Vuls を使ってないためスポンサー LT では弊社を代表して上司が主に弊社の紹介と求人してるんだよという話をしました。

元々、弊社では今年は会社の宣伝も兼ねてスポンサードできそうなものに参加していくという目標があったことと、その一例に Vuls 祭りの名前があがっていたのでスムーズにスポンサーに立候補することができました。(昨年の流れと発注を見ていてスポンサーの予算感を知っていて会社に報告しやすかったというのもあります。)

Vuls 祭りの進め方

4 月〜5 月の決起集会

神戸さんが大体の流れを決めています。今回は adachin さんをリーダーに任命して話が進み出しました。4 月に実行委員長を任命したものの 5 月に飲み会をするまで、ssmjp が Future 社をお借りし開催するタイミングで Vuls 祭りの宣伝しようということくらいしか決まりませんでした。

いや、飲み会で決まったのも会場くらいか…

adachin さんは Vuls のコントリビューターをしている方です。
Twitter : @adachin0817

決起集会後

飲み会で会場を DMM に定めた私たちは DMM を使える日の確認を行い、登壇者と[illegible]調整を行いました。Slack はログが流れてしまい過去のやりとりを探すのが大変にな[illegible]くるので GoogleDocument や Trello を使い進捗を管理していました。

各々のツールの良いところを上手く利用していたと思います。

Slack はビジネス向けチャットアプリです。
https://slack.com/
Trello はタスク管理を行うツールです。
https://trello.com/ja/home

続きは本編にて！

第 5 話

今日も残念

書いた人：togakushi
書いた人のお仕事：SIer 屋さんで IT インフラの設計構築
書いた人の日常：たいへんよくねてます

残念って？

SIer 屋さんのお仕事をしていると「どーしてこーなった！？」ってシーンによくブチ当たります。SIer ではプロジェクトに関わる関係会社が多くなりがちで、業務をこなす人材も多岐にわたりスキルもピンキリです。

そこで SIer はプロジェクトを遂行するために必要な作業を分割し、各会社に分担していきます。プロジェクトの途中から参加してくる会社もあったり、途中で抜ける会社もあったりするので、どこかが抜けたり入れ替わったりしても、最終的にはプロジェクトをなんとかするのが SIer の役目です。

基本は「誰にでも出来るように」するために工程を分けて順番を決め、作業を分割・単純化し、手順書を作り、その通りに行えば終わるレベルまで落とし込みます。

こんな感じでプロジェクトが進んでいくのですが、実際の現場に作業レベルまで落とし込まれたものについては残念なものが非常に多いです。あとの工程になればなるほど断片化が進み、全体像がぼやけます。結果を組み合わせるとチグハグしていることもよくあります。

作業レベルでの「効率の悪いやり方」「技術的に間違っているやり方」を「残念」と呼び、可能な限り修正していくお仕事の一部を紹介します。

やりたいことしかやらない残念

過去に正しかった手順は常に正しいという強迫観念でもあるのでしょうか。RHEL4 で設定していた内容をそのまま RHEL6 に適応している場面によく出会います。ディストリビュータの作法を無視して自分らの要件だけを組み込んでいる場面にもよく出くわします。

それらがアップデートのタイミングで上書きされて動かなくなったときの言い訳に「OS のバグ」と言っちゃたりします。

一度誰かが考えたことは、それがたとえ間違っていたとしてもそのまま使い回されていきます。この手法は「既存踏襲」とか呼ばれます。この手の残念は設計の段階で極力潰し

ていきます。

サーバーをコピーして作る残念

サーバーにはそれぞれ担う役割があるのが一般的です。DB サーバーや Web サーバーなど。同じ OS であれば 7 割くらいは同じ部分があります。この部分をサーバーの台数分だけ作業すると時間がかかるので、1 台作ってコピーして残りを作る手順を考えた人がいました。素晴らしい。

しかし、OS ごとにユニークであるはずものを変更していませんでした。ホスト名や IP アドレスなど、目に付くものだけしか変更していないのです。Windows なら SID がかぶっている、そんなレベルです。見つかるきっかけはだいたい前回のプロジェクトから設計を踏襲するために「実機設定確認」とか呼ばれる作業をしたときです。

そして問題が発覚した実機はだいたい運用中ですので、その問題が顕在化するまで放置されることがほとんどです。触らぬ神に祟りなしってことで触らせてくれません。

残念です。

シェルスクリプトで日付計算する残念

RHEL6 の crond で動かすシェルスクリプトで、月末を判定しているものにも出くわします。

まず、date +%Y で西暦を取得します。次に +%m で月、+%d で日を取得します。日が 30 で月が 4、6、9、11 の時は月末です。日が 31 で月が 1、3、5、7、8、12 の時は月末です。月が 2 の時はうるう年の計算をします。年が 4 で割り切れかつ、100 で割り切れない場合は 28 日が月末です。100 で割り切れた場合は、400 で割り切れるかもチェックして、割り切れた場合は月末ではありません。2000 年以降に始まったプロジェクトでは、400 で割り切れるかどうかのチェックをしていないものも存在します。400 年後はこのシステムが動いていない前提です。

こんな判定式がダラダラと書いてあるシェルスクリプトがごろごろ転がって
あっちのプロジェクトからこっちのプロジェクトにコピペして使います。そん
トは date +%m と date -d day +%m の結果を比較して、それが一致してた
判定するようにそっと書き換えてレビューを通しちゃいます。

テスト・評価が手作業な残念

設計書を作り、製造工程を越えるとテスト工程に入り
結果をチェックするのは鼻を効かせてヤバそうなと
工程なので、スケジュールが押している場合が
断も入ってきます。

続きは本編にて！

第 6 話

Ansible のプロジェクトドキュメントを Sphinx でイイ感じに管理してみる件

かいたひと：とがくし
Twitter：@togakushi

はじめに

本章では筆者が普段業務で利用している Sphinx の利用事例を紹介します。筆者は Ansible を利用する環境において、ドキュメント作成を Sphinx で統一しており、ドキュメントとコードをまとめて管理しています。

なお、Sphinx で扱う形式は HTML のみを対象としており、PDF などへの変換は考慮されていません。

環境

業務で利用している環境は **Red Hat Enterprise Linux 7**(RHEL7) であるため、Python のバージョンは 2.7 となります。もうじきサポートも終了し、マルチバイト文字の扱いにも気を使うバージョンですが、RHEL7 なのです。

本章で触れないこと

以下の項目は、本章では触れません。

- ツール (Ansible、Sphinx) 自体のインストール方法、細かな使い方
- バージョン管理
- 継続的インテグレーション
- Ansible Tower を絡めた使い方

Ansible のディレクトリ構成

Ansible のディレクトリ構成は、公式ドキュメントの “Best Practices“ に紹介のある “Alternative Directory Layout“ に近い構成を取っています[*1]。

筆者の環境では、プレイブック、ロール、インベントリを保存するディレクトリ作成し、さらにそれぞれ目的に応じたサブディレクトリを作成しています。

ロールを保存しているディレクトリ

業務の都合上、業務プロジェクトには Excel で記述された設計書が必ず存在します。ロールの構成は Excel の設計書の構成になるべく合わせます。

Excel で記述された設計書は、仕様追加や記述漏れなどで途中に新しい項目を差し込む必要が出た場合にメンテナンスコストが異様に高くなることがあります。

コストの高い修正は妥協され、情報があるべき個所に記載されずに一番最後に追記されていることがよくあります。

このような構造が目茶苦茶になっている設計書をそのままロールとして表現するとかなり歪な形になりますが、メンバーの負担軽減を優先し、まずは設計書の内容をそのままロールに書き起こします。設計書の構造が一気に変わってしまうと、プロジェクトメンバーが情報の記載された箇所を見失い、ストレスが高まる傾向になります。

設計書の内容をロールに書き起こす段階では設計書の修正を行わず、腐った設計書は時間をかけて徐々に正しい姿へ修正していきます。

プレイブックを保存しているディレクトリ

プレイブックは利用シーンと使用頻度に応じて以下のサブディレクトリに分けて保存しています。

configuration
: サーバー設定全体を行うプレイブックを保存。AP サーバーや DB サーバーなど、サーバー役割に応じたプレイブックを準備しています。設計書の内容が最初から最後まで記述されています。

onetime
: サブスクリプション登録などの 1 回実行を済ませれば 2 回目以降はもよいタスクだけを集めたプレイブックを保存。
調査のためにログを回収するだけのタスクなど、不定期に実行このディレクトリに保存しています。

update
: 運用によって変化した状況をアップデートするためconfigurationの一部であるためタグを指定してレーションの煩雑さを取り除くために必要な

続きは本編にて！

[*1] https://docs.ansible.com/ansible/latest/usa alternative-directory-layout

第 7 話

ESP-WROOM-32 用ブレイクアウトボードのステンシル作ってリフローしてみた

書いた人: けい
Twitter ID: _keihino
書いた人のお仕事：ものづくりする人を一所懸命に応援する人

はじめに

電子工作レイヤのみなさま、こんにちはこんばんはおはようございます！ 最近、ESPRESSIF (https://espressif.com/) 製の ESP-WROOM-32 モジュールが、日本でも発売されて話題になっています。

本稿では、この ESP-WROOM-32 モジュールに対応したブレイクアウトボードにクリームはんだで部品を実装するために、プラ板とレーザーカッターでステンシルを作ってみました。運用についての本ということで、レーザーカッターとリフロー炉の運用方法の紹介になります。

（※基板の作り方がある程度わかっていないとおいてけぼりですがすみません。）どこのご家庭にもある設備・部材・データ一覧

- ブレイクアウトボード 1 枚
- タミヤ プラペーパー 0.1mm 厚 1 枚
- クリームはんだ 5g くらい
- スキージ 1 個
- 竹串等 1 本
- ブレイクアウトボードに載せる部品 BOM を見ましょう
- Trotec レーザーカッター Speedy300 c60 1 台
- はんだリフロー炉 1 台
- ガーバーデータ（ランドが表示されているデータ）
- KiCad 4
- Rhinoceros 5
- Adobe Illustrator

やったこと

1. KiCad を使って、ガーバーデータから dxf データに変換する
2. Rhinocerosを使って、dxf データの曲線を統合する
3. Illustrator とレーザーカッターを使って、ステンシルをカットする
4. クリームはんだを塗る
5. 部品を置く
6. リフロー炉を使って、クリームはんだを焼く

1. KiCad を使って、ガーバーデータから dxf データに変換する

KiCad というオープンソースの回路設計・基板設計ソフトウェアを使用して、ガーバーデータを読み込み、dxf データに変換します。

1.1. GerbView でガーバーデータを読み込み

KiCad は、様々なソフトウェアの集合で構成されています。KiCad のソフトウェアのうち、GerbView を使用して、ガーバーデータを読み込みます。読み込む際は、ガーバーデータのうち、拡張子が .GTS となるものを選択して読み込んでください。.GTS ファイル は、部品面のレジストを表すデータになります。（図 7.1・図 7.2）

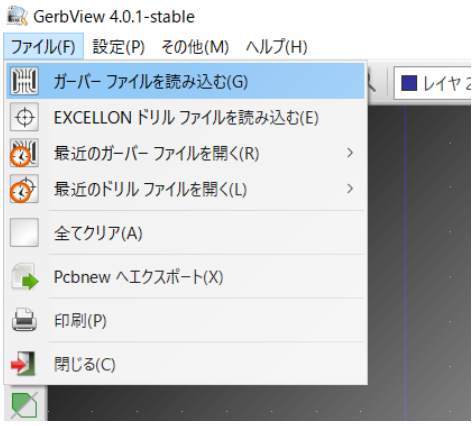


図 7.1: ガーバーデータの読み込みメニュー

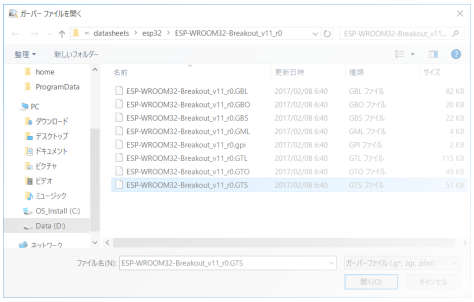


図 7.2: 同読み込みダイアログ

続きは本編にて！

第 8 話

ライダーベルトを分解してつくりかける話

かいたひと @_keihino

小さい頃、Black と Black RX を見てからというもの、平成ライダーはすべて見ている私です。#ssmjp では、おしゃべり豚野郎と名乗っております、@_keihino と申します。

さて、今回の技術書典では、ライダーベルトの分解と、その再現について書かせていただきます。

なぜライダーベルトをつくるのか

平成ライダーに入ってからというもの、変身フォームは増加の一途をたどっております。やはり玩具がないと支えられないとはいえ、現行のジオウの前のビルドに至っては、60 のベストマッチがあり、それぞれに音声がある状況。こりゃもうどうしようもないわ…レジェンドライダーもどんどん増えるのですから、そのコラボも入れると、120 を超える音声が収録される事態となっています。

その音声量を収録するためなのか、最近の変身アイテムには、認識ピンと呼ばれるスタッドが立っているプラスチック部品があり、このピンの長短、認識タイミングの組み合わせで、120 を超えるパターンを実現しています。

この認識ピンを再現すれば、もし市販の新しいおもちゃがほしい！ と言われた場合でも、そのとおり再現してあげて、音声を別録りして何らかのマイコンで出してやれば、新しく買わなくてもお茶が濁せます！ これは貧乏人には仕方がないことだ！

認識ピンのパターンは、検索すると検証されているブログ等が出てくるので、本稿では省略させていただき、パターンではなく、認識ピンを如何に再現したか、について書きます。

ライダーベルトの構成（認識ピンまわりだけ）

認識ピン

3 つのスタッドの立て方を工夫することで、120 を超えるパターンを再現しています。

今回は、ビルドドライバーに付属している、フルボトルの認識ピンを測定して再現しました。測定はノギス、再現は 3DCAD です。再現図を示します。

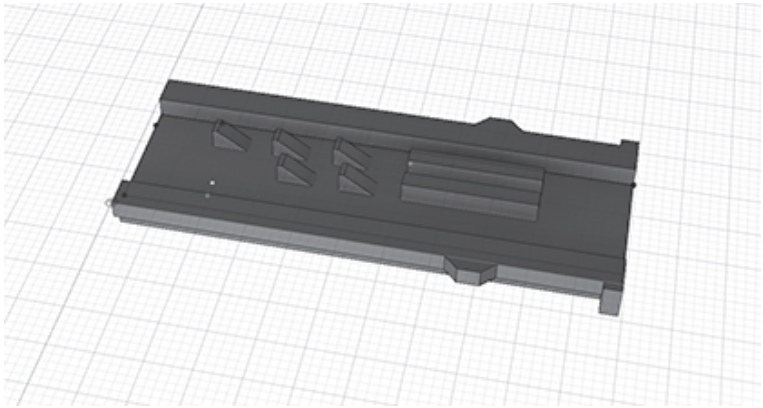


図 8.1: 3DCAD で再現したスタッド

3DCAD で起こしておくと、パターン生成も部品の表示のオンオフだけでできるため、非常に楽です。ここから 3D プリントで再現可能です。

認識ピンの受け側

電気的な話

3 つのスタッドを受ける、3 つのスリット上に、ディテクタースイッチが配置されています。

図 8.2: 3 つのスリット

続きは本編にて！

第 9 話

【未解決】Raspbian buster リリースでモノラルアンプがちゃんと使えない

かいたひと: _keihino a.k.a. うっちゃりボーイ（おしゃべり豚野郎）
Twitter: @_keihino

まえがき

どうもこんにちは、_keihino と申します。IoT 的ななんとやらをしています。今回は、Raspberry Pi Zero W と I2S モノラルアンプの話です。

Raspbian buster で hifiberry-dac のドライバどこいった

みなさま、Raspberry Pi（以下、RasPi） で I2S のアンプを使われることはよくあると思います。私もよく使っており、中でも Adafruit の MAX98357A をモノラルアンプとして使っているのですが、Raspbian のバージョンを stretch から buster に上げたところ、今まで使えていたドライバがさっぱりなくなってしまっていました。このドライバをどうやって代替するか悩んだ記録になります。

ただ、結果として、微妙に使えるようになったものの、未解決状態です。正解をご存知の方がいらっしゃいましたら、Twitter 等でご連絡をいただけると助かります。

調査環境

- ハードウェア
  - Raspberry Pi Zero W
  - Adafruit MAX98357A mono Amp
  - 8 Ω 1W 等のスピーカー（アンプにつけられるもの）
- OS
  - Raspbian 2019-07-10 版 (buster)

問題のドライバ

stretch(2019-04-08版) までは、/lib/modules/4.14.98+/kernel/sound/soc/bcm配下に、snd-soc-hifiberry-dac.ko　　というドライバが存在していたのですが、buster(2019-07-10版) ではそれがなくなりました。Raspbian の github リポジトリを追いかけていたんですが、カーネル 4.18.y 系から、snd-soc-rpi-simple-soundcardというドライバに置き換わったようです。[*1] [*2]

ところが、このドライバ、どうも動かし方が分かりません。そこで、どうにか代替させることにしました。

代替させるためには、どのドライバを使うのかを判断しなければいけません。Raspbian（というか ARM　アーキテクチャ）は DeviceTree を採用しているので、/boot/config.txt　に dtoverlay=xxx　といった感じで Device Tree Blob(dtb) を指定して、必要なデバイス設定を呼び出すように設定します。Raspbian では、/boot/overlays/　配下に、dtbo ファイルが置いてあります。このファイルの中身を見て、ドライバを探すことにしました。

■コラム: DeviceTree について

ARM　アーキテクチャの Linux では、ドライバがパラメータ違いごとにどんどんカーネルに追加されて、どんどんカーネルが大きくなっていきました。それに Linus がキレたので、DeviceTree のような仕組みが出来上がったらしいです。詳しく説明するとページを食うので詳しくはググってください。もともとは違うプロジェクトだったのが、現状 DeviceTree としてまとまった感じです。

DeviceTree では、ドライバ側は汎用的に書いておき、DeviceTree Sourceにドライバが使うハードウェアのパラメータを記述します。これによって、ドライバ自体は肥大化せずに書けるようになり、割り込みやアドレス等のパラメータは DeviceTree Source 側に追い出せます。詳しくはググってください。ページがなくなるのd（以下略

buster で音を出すまでの調査方法 (hifiberry-dac.dtbo　編)

さて、dtboファイルはあるので、ここから探しましょう。DeviceTree Blob
イルは、DeviceTree Compiler(dtc) を使って、DeviceTree Source(dts)
ことができます。/boot/overlays/max98257a.dtbo　の存在も気に
/overlays/hifiberry-dac.dtbo　　から見ていきます。

[*1] https://github.com/raspberrypi/linux/commit/12
diff-7d15d07c0ee0188707b1cf5ed7c234e3

[*2] https://github.com/raspberrypi/linux/comm
diff-7d15d07c0ee0188707b1cf5ed7c234

続きは本編にて！

第 10 話

PDCA に対する誤解と真実

はじめに

某運用系街角勉強会の主宰をしている、yakumo3 といいます。どうぞ呼び捨てにして「やくもさん」と呼んでください。本誌は#ssmjp 非公式となっていますが、面白そうだったので僕も何か書きたいと思いました。

運用と言えば「回らない PDCA」という誤解かなと思い誌面をいただきました。皆さんは PDCA と聞いて、どんな印象を持つでしょう。回らないもの、古臭いフレームワーク、意識高い人が超高速で回すもの、という印象を持っていませんか？

ここでは、PDCA の本来の姿を再度確認して、正しく認識することで僕らの業務や生活に良い影響を及ぼすことを期待します。

PDCA がやってきた

まず、PDCA は品質管理の根本的理念として導入されたことを踏まえておきましょう。1950 年にデミング博士により、「設計 → 製造 → 販売 → 調査・サービス」の 4 つが円環状になって回転するサイクル（通称デミングサークル・デミングサイクル、4 つが円環状に並ぶ図からサークル、4 つのプロセスを回す事からサイクルと呼ばれる。本文では「プロセスを回す」事についての解説をするため「サイクル」を採用しています。）が日本に持ち込まれました。博士自身は「これはシューハート博士の考え方だからシューハート・サイクルと呼ぶべきだ」と言っていたと伝わっています。このデミングサイクルは各プロセスが異なる部門、異なる職種が実行する事を想定されたサイクルでした。

このデミングサイクルを元に、日本国内で 1952 年に「設計 → 製造 → 検査・購買 → 消費者調査」というサイクルが生まれます。それが 1954 年に「企画 → 作業 → チェック → 処置」というサイクルに発展し、さらに 1959 年に管理一般へのデミングサイクルの適用として、「 計画 → 実行 → 反省 → 処置」というサイクルが生まれます。
1964 年に石川馨がこのサイクルに、「Plan → Do → Check → Action」という英語名を定義します。ここで初めて PDCA サイクルという言葉が生まれました。
この手法は「 科学的管理のいわゆる plan - do - see (1903 年) をもう少し実際的に、QC 的に考えた手順である」「昔から plan - do - see という言葉があったが、これは日本人にはむかない、see という言葉を”見る”と習っているので、やってみて眺めているだけということになりやすい」として考案したサイクルだと石川馨は言及しています。

一方、デミング博士は日本での知名度と裏腹に帰国後コンサルティングをしながらも

米国では全くの無名でした。しかし 1980年に NBC がドキュメンタリー番組「If Japan Can... Why Can't We? 」（日本にできるのに、なぜアメリカができないのか？）を放映した事により、その活動と実績が脚光を浴びます。フォード社は彼を品質管理の専門家として招聘しましたが、すでに PDCA サイクルを単なる品質管理から経営哲学にまで発展させていた博士は、問題の 85 %はマネジメントの責任だとしてコンサルティングを行い、フォード社に大きな実績を残しました。そして、全米の多くの企業から博士への講演依頼が殺到し、1990年の米国の躍進に多大な功績を残した、と伝えられています。

このように元々は戦後の日本に品質管理のサイクルとして伝来したデミングサイクルが、その初期の頃に応用として管理一般のサイクルが生まれました。品質管理と管理一般の 2 つの PDCA サイクルは TQM （ Total Quality Management ）へと発展し、やがて幅広い分野に PDCA が適用されていきます。

PDCA への誤解

さて、PDCA サイクルを用いて戦後日本は高品質・高性能な製品を量産し、高度経済成長を成し遂げたという事実がありながら、なぜ PDCA サイクルが「回らない」「古臭い」という印象をもたれているのでしょうか。そこにはバブルの崩壊と、同時期の米国の躍進 (実際はデミング博士が日本に伝えた PDCA 及び TQC ・TQM を米国流に取り入れた成果だが当時の日本はそれに気が付いてなかった) による旧来の日本手法への否定と米国方式の賛美と共に、PDCA サイクルの各プロセスの理解不足があるように感じています。これは筆者の主観です。

PDCA サイクルにおいて、Plan → Do → Check → Action の Action は Plan へと反映されなければなりません。
全ての活動は計画に則って実施されるべきであり、修正是正対策は全て計画に組み込まれる必要があります。
デミング博士自身も 1982年記したにその著書の中で、PDCA の P がいかに重要であるかを説いています。

ところが、多くの人々は Plan に戻すこと無く是正した結果を実行（ Do ）してしまいます。これには各プロセスへの「誤解」があるからと感じます。

まず PDCA サイクルのおさらいをしておきましょう。

続きは本編にて！

第 11 話

HTTP ステータスコードの話

かいたひと：yakumo3 (ssmjp 2 代目主宰)
Twitter: @yakumo3

HTTP のステータスコードは IT エンジニアなら一度は目にしたことがあると思うし、気が付いたら頻出のコードの意味を何となく覚えてる人も多いかと思います。ステータスコードを把握しておくことは Web 系のエンジニアにとっては必須ですが、そうでない人にとっても知っておくと面白いよ、と思ったのでちょっとした解説をしようと思いました。元ネタとしては同僚とのチャットでの会話に使えて便利だったのです。

同僚「今月のレポート作成変わって貰えませんか」
僕「402」
同僚「そこをなんとか…」
僕「406」
同僚「…」
僕「408」

ここでは、上記のような符丁として使われるようになると面白いなぁと思って紹介していますので、紹介している内容は厳密な用法というよりは『この数字はだいたいこんな意味』というざっくりとしたものになっていることはご了承ください。厳密なものは RFC を読んでいただくか、膨大な量の解説本が出てますのでそちらを参考にしていただくのが良いと思います。

なお、紹介しているのは RFC を基準としています。独自実装系のステータスコードも多数ありますが環境依存が強く、言ってみればマニアックすぎて普段使いしにくいので今回は割愛しています。また観点は「普段使いしやすい」という視点で見ているので、本職の Web 屋さんの使い勝手とは異なります。

HTTP ステータスコード

1xx 情報
2xx 成功
3xx リダイレクション（移動）
4xx クライアントエラー
5xx サーバエラー
の 5 種に分類されています。

1xx Informational 情報

リクエストは受け取られたけど、処理は終了していないのでエラーでも終了でもない。HTTP/1.0 では 1xx のステータスコードは使えない。

100 Continue [RFC7231]

継続。クライアントはリクエストを継続できる。サーバがリクエストの最初の部分を受け取り、まだ処理の最中な状態。
以下のような使われ方をする。

1. サーバーへ Expect: 100-continueヘッダをつけたリクエストヘッダを投げる
2. サーバーから 100 Continueが返ってくる
3. サーバーへリクエストボディを送信する

101 Switching Protocols [RFC7231]

プロトコル切替え。このステータスコードを返す時、サーバーは Upgradeヘッダーフィールドを送信して受け入れ可能なプロトコルを示す。
Websocket通信や HTTP 1.0 で届いたリクエストに対して HTTP 1.1 または TLS1.1 で通信させようとする場合に使う。

102 Processing [RFC2518]

処理中。WebDAV の拡張ステータスコード。処理が継続されて行われている状態。クライアント側はタイムアウト等で勝手に切断しちゃだめ。

103 Early Hints [RFC8297]

早期のヒント。HTTP/1.1 及び HTTP2 のリソース配信の最適化に利用する
最終的なレスポンスヘッダが確定する前に、予想されるヘッダを伝達する。Link レスポンスヘッダと組み合わせて、関連するリソースの先読み・プッシュ配信に利用する
想定されている。

2xx Success 成功

正常終了系のステータスコード。
だいたいリクエスト自体は受け付けられてる。203とか疑
スコードも使い勝手は良さそう。

200 OK [RFC7231]

OK。リクエストがサーバに届き、レスポ
れるときに使われる。

続きは本編にて！

第 12 話

とある診断員の回顧録~脆弱性診断の現場から愛をこめて~

書いた人：tigerszk

はじめに

皆様こんにちは！ 私は ssmjp 運営メンバーの一人の tigerszk というものです。

突然ですが読者の皆様は「脆弱性診断」というものをご存知でしょうか？ 脆弱性診断とは、システムやアプリケーションに含まれる脆弱性（セキュリティ上の問題点）を洗い出すための検査のことです。一般的には診断する対象、要求する内容などによって実施する診断の種類が分かれており、色々な診断があります（プラットフォーム診断、Ｗｅｂアプリケーション診断、スマートフォンアプリ診断、ペネトレーションテスト…）。

読者の方の中には、セキュリティ会社に依頼をして、自分の管理するシステムに対して脆弱性診断を実施した経験のある方がいらっしゃるかもしれません。

実は私はセキュリティエンジニアとして、この脆弱性診断が仕事の「脆弱性診断士」にかれこれ 10 年近く従事しております。今回の記事では一般的には結構マニアックだと思われるこの仕事の魅力や私が経験したエピソードなどを書かせていただきます。

脆弱性診断の仕事って面白いの？

この仕事の魅力は、単的に言ってしまえばズバリ「合法的にハッキングをできる」ということでしょう。脆弱性を見つけるってのは結構面白いんですよ！ 多分ここに面白みを感じられるかどうかがこの仕事への適正ポイントなんじゃないかと思っています。

「それじゃ、いってることが攻撃者と変わらないじゃねーか。そんなこと書くんじゃねーよ」って色んな所からマサカリが飛んできそうなのでちゃんと書いておきますが、もちろんそれだけではありません。脆弱性を見つけると結構、感謝いただけることが多かったりします。まあもちろん「余計なもの見つけやがって」とか「俺の仕事増やすな」みたいにネガティブに受け取られてしまう場合もあります。

ただ、やっぱり危険な脆弱性を発見して報告する時などは、「顧客のために診断をしていることが、ひいては社会全体のためになることであり、自分は世の中の役にたっているのだ！」という実感があります。ジョジョのポルナレフの言葉を借りるのであれば、「傷

ついた体でも勇気が湧いてくる『正しいことの白』の中に俺はいるッ[*1]」というところでしょうか。私自身実際に放置しておいたら重要なインシデントが発生するかもしれない脆弱性を見つけたことは何度もあります。そんな時に、「見つけてくれてありがとう」などの感謝の言葉をいただいたり、自分の報告した脆弱性を修正いただいたりすると、やっぱり素直にうれしいです。これもこの仕事の魅力の一つではないかと思います。

それと、結構勘違いされている方がいらっしゃいますが、残念ながら毎日華麗にハッキングをキメまくっているだけのお仕事ではありません。現実には意外と地味な作業が沢山あります。例えば、提供する診断サービスの種類によってはインフォメーションレベルの項目も含めて網羅的に洗い出さなければならなかったりするので、かなり細かいとこまでチクチク確認しなければならなかったりします。

また、顧客やシステム担当者との調整作業などが頻繁に発生するのでそういったことに対応する時間が非常に多いです。後は診断結果をまとめた報告書を作成したりもしなければならないですね。もちろん脆弱性診断をやっている全ての方がそうではなく、ポジショニングなどによっては色々違うかもしれないですが、私が知っている同業者の方々は、そういった地味な作業も含めて、コツコツしっかりとやられている真面目な方が多い印象です。

また、この仕事をやっていると多様なシステムに遭遇することができるというのも特徴の一つでしょうか。診断の依頼が来るシステムは本当に千差万別です。様々なプラットフォーム、ミドルウェア、アプリケーションフレームワークで構築・開発されたシステムと触れ合うことになると思います。そして、それと同時に世の中には摩訶不思議な珍システムが沢山あることと、思った以上に脆弱性が溢れていることを体感できるでしょう。

「哀しみを背負うことで強くなる」ここが辛いよ脆弱性診断士

そんな脆弱性診断のお仕事ですが楽しいこともある反面、正直辛いこともあります。ここでは私が過去踏んだ地雷案件について書こうかと思っています。すでに終わった昔話ですので笑い話として読んでいただければ幸いです。

脆弱性診断は基本的には診断対象に影響をあたえない前提で実施します。トラブルにならないためにしっかりシステム担当者と事前調整を行ったり、稼働中のシステムに影響
与えないように吟味した項目だけを診断したりします。しかしながら、注意してい
はりトラブルを避けられない場合もあります。過去 ssmjp で発表していますが
断キャリアの中での最大級の地雷は、「本番顧客サイトのパスワードを全部
まった」という話です。

ことの顛末については参考資料のスライド[*2]に詳しく記載して
ればご覧ください。最近耳にしたのですが、ありがたいこと
利用いただいている方もいらっしゃるとのことで、私の
ば幸いです。

続きは本編にて！

*1 ジョジョの奇妙な冒険第 27 巻を参照
*2 参考資料：「とある診断員と SQL インジェク
sql-35102177

第 13 話

PHP でも Raspberry Pi がしたい！

かいたひと：東雲翡陽（PHP カンファレンス実行委員長, CONBU, 他）
Twitter：@H_Shinonome facebook：hiyou.shinonome

はじめに

この記事は 2019/03/29-31 に行われた PHPerKaigi 2019[*1]での発表「PHP でも Raspberry Pi がしたい」[*2]のセッション内容を改稿・追記したものです。発表当時と内容が若干異なりますが、ご了承ください。

Raspberry Pi 概要

Raspberry Pi とは

Raspberry Pi（ラズベリーパイ）は、英国ラズベリーパイ財団によって開発されているシングルボードコンピュータです。2012 年より発売され、今年で 7 周年を迎えます。世界累計販売台数は 2000 万台を超え、非常に注目されているコンピューティングモジュールです。CPU として ARM ベースの **SoC**（System on a Chip）を搭載しています。ARM ベースのため、消費電力が少なく、標準ではヒートシンクもついていないほど比較的発熱も少ないです。そして、Compute Module というホームファクタ以外は USB コネクタを用いた +5V の給電で動作します。一番安価な Zero が約 600 円〜、最新の 3 Model B+ が約 5000 円〜と非常に安価に購入することができます。また、GPIO と呼ばれる外部接続用ピンヘッダがあり、それを用いて電子回路と組み合わせることで (理論上は) なんでも操作が可能になります。

[*1] https://phperkaigi.jp/2019/
[*2] https://fortee.jp/phperkaigi-2019/proposal/6ff7c81a-e6d2-407c-af2b-bd486b7589c9

Raspberry Pi 7 周年イベント

2019年が Raspberry Pi 7 周年ということで、Raspberry Jam Big Birthday Weekend 2019 というイベント[*3]が世界各国で行われました。開催日は 2019/03/02-03 で、各国のユーザーグループ等が Raspberry Pi の 7 周年を祝いました。日本のユーザーグループである Japanese Raspberry Pi Users Group でも 2019/03/03にイベントが開催されました。[*4]

GPIO （General Purpose Input / Output ）

さて実際にどのように電子回路を操作するのか、簡単にご紹介します。概要でも触れた GPIO ですが、Model A が 26pin、Model B と Zero は 40pin のピンヘッダで構成されています。3.3V 電源、5V 電源、Ground といった電源系統のピン以外は、SoC に直接配線されていますので、間違った回路をつなぐと最悪の場合 SoC が死にます。回路をつなぐ場合は使うピンヘッダが間違っていないかを慎重に確かめ、電源を最後につなぐようにしましょう。なお、ピンヘッダの用途は下記の表のとおりです。

Raspberry Pi 3 Model B のヘッダ例

```
+-----+-----+---------+---Pi 3---+---------+-----+-----+
| BCM | wPi |   Name  | Physical | Name    | wPi | BCM |
+-----+-----+---------+----++----+---------+-----+-----+
|     |     |    3.3v |  1 || 2  | 5v      |     |     |
|   2 |   8 |   SDA.1 |  3 || 4  | 5v      |     |     |
|   3 |   9 |   SCL.1 |  5 || 6  | 0v      |     |     |
|   4 |   7 | GPIO. 7 |  7 || 8  | TxD     | 15  | 14  |
|     |     |      0v |  9 || 10 | RxD     | 16  | 15  |
|  17 |   0 | GPIO. 0 | 11 || 12 | GPIO. 1 | 1   | 18  |
|  27 |   2 | GPIO. 2 | 13 || 14 | 0v      |     |     |
|  22 |   3 | GPIO. 3 | 15 || 16 | GPIO. 4 | 4   | 23  |
|     |     |    3.3v | 17 || 18 | GPIO. 5 | 5   | 24  |
|  10 |  12 |    MOSI | 19 || 20 | 0v      |     |     |
|   9 |  13 |    MISO | 21 || 22 | GPIO. 6 | 6   | 25  |
|  11 |  14 |    SCLK | 23 || 24 | CE0     | 10  | 8   |
|     |     |      0v | 25 || 26 | CE1     | 11  | 7   |
|   0 |  30 |   SDA.0 | 27 || 28 | SCL.0   | 31  | 1   |
|   5 |  21 | GPIO.21 | 29 || 30 | 0v      |     |     |
|   6 |  22 | GPIO.22 | 31 || 32 | GPIO.26 | 26  | 12  |
|  13 |  23 | GPIO.23 | 33 || 34 | 0v      |     |     |
|  19 |  24 | GPIO.24 | 35 || 36 | GPIO.27 | 27  | 16  |
|  26 |  25 | GPIO.25 | 37 || 38 | GPIO.28 | 28  | 20  |
|     |     |      0v | 39 || 40 | GPIO.29 | 29  | 21  |
+-----+-----+---------+----++----+---------+-----+-----+
```

この表は下記のコマンドで取得ができます。

```
$ gpio readall
```

続きは本編にて！

[*3] https://www.raspberrypi.org/jam/big-birthd
[*4] https://www.raspi.jp/2019/01/raspberry-

第 14 話

ひとり NOC をやった話。あるいはインターネット接続を提供するということ (1)

かいたひと：東雲翡陽
（PHP カンファレンス 2019 実行委員長
CONBU コアメンバー, 他）
Twitter: @H_Shinonome
Facebook: hiyou.shinonome

はじめに

最近、各種カンファレンスにおいて、無線 LAN を開放して参加者にインターネットアクセスを提供する主催者が増えています。4G/LTE において月あたりの通信量に制限があるのが一般的な昨今、SNS に投稿して場の雰囲気を共有できたり、リモートで仕事をできたりと、インターネットに隣接する我々 IT 業界の人間にとってはありがたい状態です。しかし、例えばいざカンファレンス主催者が参加者に無線 LAN を提供しようとしても何から始めればよいのか、何を用意すればよいのか、気を付けなければいけないポイントは何なのか、非常に分かりにくいものと思います。というのも一般的な無線 LAN 施工は、例えばオフィスの天井にアクセスポイント（以下、AP）を固定したり、利用者とその端末が管理、統一されていたりと、環境の構築が行いやすい条件がそろっています。しかし、カンファレンスにおいてそうそういい条件がそろうとは限りません。そのようなカンファレンスにおける無線 LAN 構築と、インターネット接続の提供について、できるだけ安価かつ確実に構築を行えるよう自分の持っているノウハウを公開したいと思っています。

はじめに「インターネットアクセスの提供にあたってなにを考えたらよいのか」を今回はお送りいたします。

インターネットアクセスを提供するということ

単純にインターネットへの通信をできるようにするためには、下記のような物が必要だと思い浮かぶ方も多いと思います。

- 例えばフレッツ等の回線を契約
- ブロードバンドルーターを接続

- ルーターが無線 LAN を提供する or LAN ケーブルで PC を接続する

一般のご家庭でインターネットを使う為によく用意される構成ですね。では、カンファレンスにおける無線 LAN 提供と何が違うのかを掘り下げてみましょう。

- 利用人数
- 利用者
- 端末数
- 端末種別
- 利用帯域

まず、この 5 つは明らかに家庭やオフィスと違うものです。利用人数は、大きいカンファレンスでは全体で千人以上になる場合もあります。家庭用ブロードバンドルーターでは、多くても 20 人がせいぜいでしょう。そして、利用者もオフィスや家庭であればその中にいる人のみが利用しますが、カンファレンスは登録こそありますが、基本的には不特定多数への提供になります。また、端末数や端末種別も一人で PC とスマホ、場合によってはスマホを複数台、そしてそれぞれの無線 LAN の対応周波数が違う、といったと IT 業界ならではの数と種類になります。利用帯域も同様です。オフィスでは、ずっと重いインターネットアクセスをする、例えば YouTube を見続けるといったようなことはほとんどないかと思いますが、カンファレンスでは SNS にアクセスしたり、自分がいるところと違う会場のストリーミング配信を閲覧したりと比較的重めのアクセスが発生します。ハンズオン等をやると GitHub からコードを持ってくるなど、同時多発的なアクセスも発生します。不特定多数の参加者へ提供するにあたり、気を付けなければいけないポイントもあります。例えば無線 LAN 接続に対価を求めると電気通信事業者としての対応が必要となるため、一般的にカンファレンスでの無線 LAN 接続において追加費用を参加者からもらうことはできません。また、その通信回線において非合法なアクセスがあった場合、できる限りの証跡提出を求められる可能性もあります。

インターネットアクセスを提供するために必要な機能

基本的にインターネット接続を提供するために必要なものはブロードバンドルーターと一緒です。ブロードバンドルーターは一般的に下記の機能を提供しています。

- インターネット回線終端
- NAPT(IP　マスカレード)
- DNS キャッシュサーバー
- DHCP サーバー

インターネット回線終端

ブロードバンドルーターはその機能としてフレッツ回線
り当てられるグローバル IP(回線種別によってはプラ
端された IP によって利用者はインターネットへの

続きは本編にて！

第 15 話

時間のないエンジニアに贈る料理の運用の話

書いた人：Mary　（Twitter @mary_tuba)

こんにちは！　Mary（メアリー）と申します。普段はどっかの会社でインフラエンジニアをやっています。そこでインフラ周りの運用のお話をするかと思えば…料理の運用のお話をしようと思います。どうして料理の運用をしようと思ったかって？

それは ssmjp の中でも料理担当と言われていたり、一応兼業主婦？ であったりもするからなのです。共働きの家庭ならではの運用はまだまだたくさんありますが、今回は料理の運用の話をしたいと思います。

今回（次回はあるのかしら…）の題材は

- 圧力鍋の話
- 圧力鍋料理のレシピ
- 簡単レシピ（電子レンジ系）
- 簡単レシピ（包丁を使わないレシピ）

のこの 4 点でお話したいと思います。

圧力鍋の話

圧力鍋。おそらく料理をしない人でも一度は聞いたことがあるでしょう（爆弾とかね)。ここではなぜ圧力鍋を使えば時短になるのか、おいしく食べ物ができるのか…をお伝えしたいと思います。

さて、ここで便利な Wikipedia 先生の文章を引用しましょう。

> 圧力鍋（あつりょくなべ、英: Pressure cooker[1]）とは圧力調整機構が付いた鍋[2]。空気や液体が逃げないように密封した容器を加熱し、大気圧以上の圧力を加えて（加圧)、封入した液体の沸点を高めることで、食材を通常より高い温度と圧力の下で、比較的短時間でより美味しく調理することができる調理器具である。圧力釜（あつりょくがま）とも呼ばれる [2]。
> 圧力調整には通常金属製の錘などが使われることが多い。加圧源の殆ど [3] に水の蒸気圧を利用するため、水分を伴わない調理には向かない。（Wikipeida：圧力鍋より）

おそらく Wikipedia 先生を読んでいただければ大体のことはわかるとはおもいますが、ここではざっと説明します。

通常、私たちが住んでいる平地の気圧は 1 気圧です。1 気圧の場合、水の沸点は大体 100

℃です。
なお、気圧が高くなればなるほど沸騰する温度は高くなっていきます。通常よく売っている通常圧力鍋の場合は大体 1.8 気圧まで加圧することができます。
その時の沸点は約 118℃になります。なお、高圧圧力鍋では約 2.4 気圧まで上げることができます。
その際は水の沸点は 130℃近くになります（富士山の山頂では逆に気圧が下がるので、水の沸点は下がります）。水の沸点を上げると、高温で一気に調理をすることができるようになります。そのことにより実際に鍋を火にかけている時間が少なくて済むことから、時短でご飯を作ることができる…と言われています。
また、圧力鍋は火にかけている時間だけが調理時間ではありません。加圧されている時間はずっと調理時間です（加圧中の鍋の中の水分の温度は 100℃以上です）。
そういう意味でも、加圧（火にかけ、蒸気が出ている状態）をしている時間だけガスコンロにかけておけばいいので、加圧時間が終わったらコンロから圧力鍋を下ろしてしまっても構いません。
もちろん目を離して他の事をしていても大丈夫ですし、空いたコンロでもう一品料理を作ることも可能です。圧力鍋を開けるのは圧が完全に下がってからなので、それまではフリータイムです。
もちろん、ガスや IH ヒーターの使用時間も削減することができます。
　時間が短くて、調理に使う熱エネルギーの量も少なくて済む…。とてもいいことづくめだと思いますが、いくつかデメリットも記載しておきます。
　一つ目は掃除がめんどくさいです。Wikipedia に書いてあるように、鍋を密閉する必要があります。
となると、パッキンやおもりなど、部品が多いため洗い物が少々増えます（簡単なタイプもあるそうですが…）。
　二つ目は鍋が重いです。そこそこの圧力をかけるということは鍋が丈夫である必要があります。
なので、鍋自体が厚手に作られていたり、普通の鍋にはない安全機構などがついていたりします。そのために普通の鍋に比べて重めの製品が多いです。
　三つ目は恐怖心です。圧をかけるので鍋がシュッシュッ音を立てて蒸気を吐き出します。
慣れてしまえば問題ないのですが、慣れるまでは少し怖いと思います。
　四つ目は比較的高価なことです。安い鍋でも 6000円位から。通常よく売られている
のは 1 万円前後のものが多いです。
　なお、私が自宅で利用している圧力鍋は高圧圧力鍋といわれるタイプのもの
す。
高圧鍋を利用する前は通常圧力の圧力を利用していましたが、高圧鍋に
間が 1/3～1/5 にも短くなったため私はすっかり高圧鍋の虜です。
ですが、最初から高圧鍋を利用すると、おそらく怖いと思います
力鍋あたりから入ることをお勧めいたします。
　便利な圧力鍋、是非利用してみてください。

続きは本編にて！

第 16 話

料理の運用の話

かいたひと：mary（CONBU、イベントスタッフ、週末演奏家)
Twitter: @mary_tuba facebook: aiko.miyairi

はじめに

ssmjp の料理担当をしております。料理を作るのが好き、得意なこともあり、ssmjp のみなさまから「料理に関する運用」の話をして欲しいと言われたため、今回執筆することにいたしました。確か、前も ssmjp の本で料理ネタを書いたことがあるので、2 回目になります。

今回は、

- もっと気楽に自炊をしよう
- 楽になる料理の前処理
- 料理の適量とは
- おすすめの本やフォローすべき人のご紹介

の 4 項目を描きたいと思います。

もっと気楽に自炊をしよう

ssmjp のみなさまが~~入り浸る~~集まる Slack にて、料理のお悩みはないですか？ と聞いて見たところ

> フライパンを出して卵を割って目玉焼き作るのすらしんどいときでも気力を出す方法みたいなのが知りたいかも……。

とお悩みお悩みをいただきました。わかります。私もその気持ちになる時、あります。

結論：そんな時はコンビニのちょい足しや、冷凍食品に頼りましょう。

自炊というと、しっかり何かを作らないとダメ、ちゃんと 1 汁 3 菜にしなきゃ……と思う人が多いと思います。

ですが、そこまで頑張る必要がないと私は思っています。自炊は家でご飯を食べることと割り切り、コンビニやスーパーのお惣菜に冷凍食品のブロッコリーをチンして添えるなどでも十分自炊だと私は思います。

私の家の冷蔵庫には、冷凍チャーハンとコンビニの冷凍つけ麺、冷凍ブロッコリーが常備されています。どれもとても便利なのです。ちょっとお腹が空いた時に冷凍チャーハン

を電子レンジでチン。そのあとに冷凍ブロッコリーもチン。この二つを一つのお皿に添えて、ブロッコリーに適当なドレッシングやマヨネーズをかければもう一品です。

冷凍つけ麺も同様です。具が一緒に入っているものが多いため、これ一つ温めれば立派にご飯です。コンビニやスーパーでお惣菜を買ってきましょう。それを家で温めて食べる、そこに冷凍野菜を入れて食べたら？

この行為だけでも十分自炊です。チョイ足しは正義。冷凍ほうれん草もなかなか使えます。そのままバターで炒めればほうれん草のバターソテーのできあがり。セブンイレブン等で売っている肉入りカット野菜等も結構使えます。これも温めて何かに投入すれば十分一品できますから。

ただし、冷凍食品を使う時注意として、解凍方法だけはきちんと守りましょう。解凍方法間違えると、一気に不味くなるものが増えます。ちゃんとパッケージの後ろを読んで解凍、調理をしてくださいね。

料理研究家の土井善晴さんは Twitter (@doiyoshiharu) にて、具沢山のお味噌汁とご飯だけでも十分自炊です。頑張る必要はないといった趣旨のツイートをよくされています。

そう、自炊は頑張らなくて良いのです。頑張りたい時に頑張りましょう。

楽になる料理の前処理

こちらも ssmjp の Slack からの質問です。

> 時短のための前処理教えてほしいです。野菜とか肉とか前処理しておけば長期保存できて且つ調理時間も短くなるのでは？ 冷凍庫から出した塊のママでも低温で煮ることで「鶏胸肉の焼豚風」みたいなのはできるけど、毎回それになっちゃうという悩み。

パターンとして

- 冷凍する
- 加工して冷蔵する（冷凍しない）

の 2 つが考えられると思っています。

冷凍するは超定番ですが、そのまま冷凍してしまうと次に使う時にかなりめんどう
いです。お悩みにあるように塊のまま煮る的な調理方法しかなくなってしまう。

なので、私はいつも一食分に小分けにして冷凍するか、一度調理をしてそ
しています。

小分けにする場合は、小さいジップロックが大活躍しています。そ
にポーン！ です。日付ぐらいはマジックで書いておくことをお勧
次使う際には解凍も早いので時短になります。

一度調理をして冷凍する場合は、味の濃いものがお勧め
は、ブリの照り焼き。お酒、お醤油、お砂糖を１：１
れにブリの切り身を入れてそのまま冷凍。使う際に
う。経験上、お肉も下味をつけた状態で冷凍す
の場合は解凍せずにそのまま焼けます。

続きは本編にて！

第 17 話

RAID2 は何故使われないのか

はじめに

「真のエンジニアになろう」というスローガンの下、そのヒントになりそうな小ネタを幾つかお話しようと思う。これはその第 1 回。
運用エンジニアの業務領域がどこまでなのかはケースバイケースだと思うが、手順書通りに正確に作業するだけではエンジニアとは呼べないということに異論はないだろう。エンジニアは「エンジニアリング」によってものを作ったり問題を解決したりする人のことなので、その「エンジニアリング」する力を鍛えるエクササイズだと思ってお読みいただきたい。お話するネタは運用とは直接関係ないように見えるかもしれないが、コンピュータやネットワークの基礎的な技術について採り上げているので、広く応用していただけるはずだ。

RAID

HDD は壊れる。データを保存する装置である HDD は壊れたらとっても困るが、形あるものはいつか壊れるものだ。だから壊れてもデータは失われないように色々と策を講じるのであり、その端的な例が RAID だろう。最近はクラウドストレージなんかも手軽になってきたが HDD も暫くは使われ続けるはずだし、内部的に磁気ディスクが半導体メモリに置き換わっても、HDD のようなパーツは相変わらず使われ、その構成上 RAID などの対応策が必要だという点は変わらないと思う。
まぁ、ユーザから意識されない装置内部の進化によって MTBF が劇的に延び、見かけ上 RAID というテクニックが消えてしまうということはあるかもしれない。

製品例が一つもない、机上だけの規格

RAID は 1988 年の UCB の論文「A Case for Redundant Arrays of Inexpensive Disks」に由来し、その論文では RAID1 から RAID5 までが提唱されているが RAID2 は商品化された例がない。ちなみに RAID4 はパリティドライブだけを高速化する SPD-RAID4 や（もはや RAID4 とは呼べないくらい特殊な技術を使っているが）NetApp 社の製品があるし、RAID3 はシーケンシャルアクセスに強くレイテンシが低いので今でもストリーミングなどで使われ続けている。

データは 0 と 1 とでできている

ところで、データ (デジタルデータ) というのは 0 と 1 で表現されていて……なんて思っているかもしれないが、実際にそれを処理するときにはもうひとつ重要な情報が必要になる。
例えば RAID1 の場合、2 つのドライブに同じデータを書いておけば片方が壊れてももう片方からデータが取り出せるので、ドライブ 1 台までなら壊れても大丈夫ということになっている。これはつまり「こっちのドライブが壊れている」ということがハッキリしているからこそ「あっちのドライブのデータは正常だ」と見なせるのである。データを読み出すときに返される情報が 0 か 1 かの 2 種類だけだったらエラーが混ざっていても判らない。「エラー」という 3 つめの状態の存在がとても重要なのだ。

RAID2

RAID2 は他の RAID 方式と比較して、ドライブ 7 台構成で実効容量は 4 台分しか無いとか、ハミングコードの計算のためパフォーマンスが低いとかが取沙汰されることが多いが、肝心なのは「ハミングコードを使っているので間違っているやつが居たらそれが誰だか判る」ということだ。0 と 1 だけのデータの中に実はエラーによって値が間違っている箇所がある。という場合でも、間違っているデータの位置を特定し修正することができる。他方の RAID1,3,4,5 ではエラーを正直に報告しないやつが居てもそれが誰だかは判らない（その場合、エラーがあるかどうかは判るが修正できない）。
しかし、論文が発表された 1988年当時でさえエラーがあるにも関わらず知らん顔して適当なデータを返してくるような HDD は無かったのである。つまり「間違いなく 1」「間違いなく 0」「間違ってる」の 3 種類をちゃんと返してくれる HDD が普通に手に入るのだから RAID2 は要らない。
余談になるが情報理論的に言うと「エラー」というステータスをもたない 0 と 1 の 2 種類だけのデータの場合、ハミングコードは最低でも 2bit までの誤り検知と 1bit までの誤り訂正ができる。RAID3, 4, 5 で使われているパリティでは、1bit の誤り検知しかできない。HDD は 0 と 1 だけでなく「エラー」も返せるので、エラーが発生した場合はその位置を特定できる。だからパリティだけで誤り訂正ができるのだが、これが例えばシリア
通信となると 0 か 1 かしか送れないので、ハミングコードよりも更に複雑な誤り訂
を用い、修正しきれない場合は再送させることで確実性を高めている。

続きは本編にて！

第 18 話

“ネットマスク”ってなに？

はじめに

「真のエンジニアになろう」というスローガンの下、そのヒントになりそうな小ネタを幾つかお話しようと思う。これはその第 2 回。

エンジニアとは“エンジニアリング”によってものを作ったり問題を解決したりする人のことだ。その“エンジニアリング”する力を鍛えるエクササイズだと思ってお読みいただきたい。

自分とは直接関係ない話題のように見えるかもしれないが、コンピュータやネットワークの基礎的な技術について採り上げているし、その「考え方」は広く応用していただけるはず。

ネットマスク

ネットワークの設定をするときに、IP(v4) アドレスやデフォルトルーター (デフォルトゲートウェイ) などと一緒に設定するアレだ。なんだかよくわからないけど指定されたとおりに入力してるって人が多いだろうし、ネットワークエンジニアでも、半ばパブロフの犬的に『クラス C を 2 分割するなら/25 で 255.255.255.128』なんて九九の様に深く考えなくても反射的に出てしまう、そんな程度のものだろう。

“ネットマスク”ってなに?

多くの場合「ネットマスクって何？」と聞かれて答えるのは前述のような“使い方”であって“使い道”ではないと思う。「なんのためにあるのか？」と聞かれると『サブネットに分割するため』というひとがいるがそれは正しくない。明示的に入力していなくてもネットマスクは常に何らかの値が指定されているからだ。(サブネットに分割するときはデフォルト値から変更する必要がある。というのが正しい)

なんのためにあるのか？ もうすこし掘り下げて「なににどう使われているのか？」という質問にすると、ネットワークエンジニアでも明確に答えられる人は多くない。

そもそも、エンジニアはいろんなところでいろんな設定をしていると思うが、各々の設定値が何のためにどう利用されているのかをちゃんと知っているという人は決して多くないのではないかと思う (笑)。何かを参考になんとなく設定して動いたから OK って人多いよねぇ…。ネットマスクなんてありふれたプリミティブな設定値でさえもそうなのだから (ry

ネットマスクの「設定値」は、何の処理にどのように使われているのか？

ネットマスクは、送信するときに相手がルーターの先（別ネットワーク）に居るのか自分と同じネットワークに居るのかを判定するために使われている。受信するときは使われない。同一ネットワークに居るときはそのままパケットを送信すればいいが、別ネットワークに居るときはルータに中継してもらわなければならない。その区別をしているだけ。具体的には

（相手の IP アドレス XOR 自分の IP アドレス）AND ネットマスク ＝ 0

ならば同一ネットワークに居る。

せめて、エンジニアらしく

答えは出たけど、これで満足しちゃダメだよね。エンジニア的視点で少し掘り下げてみよう。

例えば、間違って設定したらどういうことが起こるのか？

かんたんに言うと、通信できる相手とできない相手が出てきたり、こちらから通信するとつながるのに向こうからは繋げられないとか、その逆のこととかが起こる。発生しうる現象を全て挙げるのは結構難しくて、ルーターの設定によっても動作が変わったりする。（なぜそうなるのかは紙面の都合で省略するのでそれぞれで考えてみてほしい）

だから、実際にネットマスクの設定ミスでトラブルが起こったりすると、その原因に気づけないことも多い。（いまどき手で IP アドレス設定するなんてあんまないかもだけど）

トラブルシューティングのときに「もしここを間違ったていたらこうなるはず」ということをたくさん知っていれば（またはその場で考え出せれば）原因の絞り込みに大いに役立つはず。

そして、間違って設定しているものを検出・修正する方法も重要だと気づくと思う。発見的統制と言われているアレだ。また、予防的統制はどの段階でどのような対策が効果的なのかも見えてくる。「正しく設定されている」ことを、設定値を目で見て確認するだけでなく、別の方法でも点検できないといけないだろうし、本番稼働後に無停止でやらなければならないかもしれない。我らが技師長が「こんなこともあろうかと」言ってたのを思い出そう。（注：実際には言ってない）

仮説検証の PDCA サイクルを回そう

ネットマスクひとつでもこれだけ“エンジニアリング”できるのだから、普段の意識や探究心が如何に重要で、改善のヒントをつかっていただけるのではないだろうか。

今回はたまたま“ネットマスク”だったが、[illegible]れによってどんなトラブルが起こる可能性[illegible]

続きは本編にて！

第 19 話

"ネットマスク"ってなに?（つづき）

はじめに

「真のエンジニアになろう」というスローガンの下、そのヒントになりそうな小ネタを幾つかお話ししようと思う。これはその第 3 回。
エンジニアとは「エンジニアリング」によってものを作ったり問題を解決したりする人のことだ。その「エンジニアリング」する力を鍛えるエクササイズだと思ってお読みいただきたい。
読者自身とは直接関係ない話題のように思われるかもしれないが、コンピュータやネットワークの基礎的な技術について採り上げているし、その「考え方」は広く応用していただけるはず。

前回までのおさらい

第 1 回は RAID2 についてお話しした。RAID2 が何故使われないのかを考察することから、歴史や成り立ちを知る事で理解が深まり、色々な技術や製品を理解する事にも役立つだろうということを示した。
第 2 回はネットマスクについてお話しした。基本的なことにもかかわらず正しく理解されていないと思われることから、間違って設定したらどうなるかを例に、仮説検証型の PDCA を回すことでエンジニアリング能力を磨こうと啓発した。

"ネットマスク"ってなに？（つづき）

第 3 回はネットマスクの例を汎化してみようとおもう。
前回のまとめでは次のような事について触れた：

- 何故その設定が必要なのか？
- 何故その値を設定するのか？
- どうやって設定するのか？
- 間違ったらなにが起きるのか？
- 間違っていないことをどうやって確認するのか？
- 間違っていた場合、どうやって検知するのか？
- 間違っていた場合、どうやって修正するのか？
- 予防・再発防止はどのようにすればよいか？
- 正しく統制されていることをどのように確認するのか？

ほんのちょっとした事であっても、ネットマスクのように間違って設定した時の影響や挙動が複雑で発見しづらいものもある。ちゃんとやろうとしたら考えなければならないことは多い。
ネットマスクの場合は普段それほど意識しなくてもあまり間違えることはなさそうだが（だから適当にやっちゃってるよね）、パターン化して適用することで手間も減り、リスクも減らせるということに気づいて欲しい。一度ちゃんとやっておけばそのあとずっと使えるいいツールになるだろう。

設定項目の管理にはどんなことが必要か

先に挙げた項目をひとつずつ見ていこう。

何故その設定が必要なのか？

その設定によってシステムはどのように振る舞いを変えるのか知っておく必要がある。デフォルトから変更する必要がない場合でも、何故変更する必要が無いのかを知っていた方がいい。
設定がどこに影響するのかも意識しよう。直接的に振る舞いを変えるだけでなく、間接的に参照されたり影響を受けるものがあるかもしれない。

何故その値を設定するのか？

まずは設定値の型を意識しよう。数値なのか文字列なのか、ラベル（キー）なのかポインタなのか、配列なのかオブジェクトなのか。
次に、システムはその値をどの様に取り扱うのか考えよう。例えばネットマスクなら、設定値は送信先がルータを介して通信するのか、直接到達できるのかを判別するのに使われていて受信するときには使われない。
そこまで分かっていれば、期待した動作を得るためにはどの様な値を設定しなければならないかは自ずと分かるはずだ。
IP アドレスで書くのかホスト名で書くのかによる動作の違いなどについても、たとえ結果は同じでも、そこに至る動作の違いで問題が起こる場所が変わってくる。

どうやって設定するのか？

設定ファイルに記述するだけでなく、コマンドで指定したり、DHCP や LDA
問い合わせて設定される場合もある。
設定はいつ読み込まれて効力を発揮するのか。起動時だけに読み込ま
動作中に変更されるものもあるし、障害などを検知して自動的に
設定を変更する時にシステムを止める（再起動する）必要があ

間違ったらなにが起こるのか？

間違った設定をしたらどうなるのかを知ること
やってみるべきだ。エラーになって起動しな
一見正常に動作している様に見えるという

続きは本編にて！

第 20 話

真のエンジニアになるには

はじめに

「真のエンジニアになろう」というスローガンの下、そのヒントになりそうな小ネタを幾つかお話ししようと思う。これはその第 4 回。
エンジニアとは"エンジニアリング"によってものを作ったり問題を解決したりする人のことだ。その"エンジニアリング"する力を鍛えるエクササイズだと思ってお読みいただきたい。

前回までのおさらい
第 1 回は RAID2 についてお話しした。RAID2 が何故使われないのかを考察することから、歴史や成り立ちを知ることで理解が深まり、さまざまな技術や製品等の学習にも役立つだろうということを示した。
第 2 回はネットマスクについてお話しした。基本的なことにもかかわらず正しく理解されていないと思われることから、間違って設定したらどうなるかを例に、仮説検証型の PDCA を回すことでエンジニアリング能力を磨こうと啓発した。
第 3 回はネットマスクの例を汎化しライフサイクルマネジメントへアプローチするヒントを示した。

第 4 回のネタは？
頭の体操的なものが続いたので、ちょっと毛色を変えてみる。

真のエンジニアになるには

筆者がこれまで出会ってきた各界の「真のエンジニア」に共通するものを挙げ、真のエンジニアがどんな振る舞いをしているのか、なぜそうするのか、どうやってそこに辿り着いたのかを考えてみたい。
時間と紙面の都合で主要なものをかいつまんでお話しするが、こういうのは深掘りすればたくさん出てくるし、切り口によって表現や見え方も変わってくるので、サラッと読んで「ふ〜ん」って思っていただければ良い。あまりダラダラ書くのも何なので「エンジニアじゃないけどすごい人」にも当てはまるようなことはあえて書かないが、そこは誤解なきよう。
お約束だが、当記事は「個人の感想です」(笑)

エンジニア的思考

論理的思考はもはやエンジニアだけに求められるものではない。逆に言えば、どんな仕事にもエンジニア的な要素は多少はあり、むしろその割合は増えてきているのではないかと思う。そして、真のエンジニアたちにはそれを超えるなにかがある。例えば、

■それはそれ。これはこれ。 真のエンジニアたちは、複雑に絡み合った問題を解き解(ときほぐし)して複数の単純で独立な問題に置き換えたうえで、それらを同時に考えることができる。難しい問題に出会ったとき、冷静かつ論理的に考えるというのは実際は非常に難しく、だからこそ真のエンジニアたちは常に自問自答しているし、自分の外(そと)にモノサシを置いている。

■常に複数の候補を持つ　真のエンジニアは答えを一つにしない。最適な答えが一つ見つかったからといって他の答えを探すのをやめない。0 か 1 かではなく確率的に考えることができるし、問題に対して複数の視点からアプローチする。制限事項やリスクに敏感で「こんなこともあろうかと」と常に言えるようにしている。

■己を知る 「彼を知り己を知れば百戦して殆(危)うからず」という諺はあえて説明するまでもないが、真のエンジニアは、自分の実力・性質・状態を客観的にちゃんと把握している。それによって固定観念や根拠のないカン、思い込みなどを排除できるし、自分自身の行動や思考をトレースし直して抽象化することで学習効率を上げている。更に、他人に成り代わって考えることで、その人の経験を盗むことまでできる人もいる。
そして、出来ないことはやらない。出来ないことが無いように日頃から研鑽を怠らない。普段の行いが「ぶっつけ本番」を回避し、見通しを良くしている。これは研究者とは違うエンジニアならではの考え方ではないかと思っている。

自分自身をエンジニアリングする

いろいろ思い浮かべ、分類したり並べなおしたり言い換えてみたりして、そのうちいくつかをここに書いてみたわけだが、結局のところ、ひとことでいうなら「真のエンジニアは自分自身をエンジニアリングし続けている」ということになると思う。なんか普通の結論だなぁ。だけど、ちゃんと出来ている人はゼロ。「存在しない」とは言わないけれど
だとすると、自分自身をエンジニアリングし続けた行き着く先は「自分を超える
アを育てる」ということになるだろう。これは単なるボヤキだが、私が若か
「人を育てられる人」が減っているという話はよくあった。年を追うごと
る気がする。

初出: 20
かい

続きは本編にて！

第 21 話

ワインとコーヒーの違い（ふたたび）

ワインとコーヒーの違い

2018 年 1 月の **ssmjp** で LT したネタだが、5 分 LT だったので『よくわからない』という声が多かったようだ。あえてここで再びお話してみようと思う。

価格決定プロセスにみるワインとコーヒーの違い

ワインは歴史があり、生産者と消費者が近く、生産者もまた消費者である。コーヒーはワインに比べて歴史が浅く、生産者はコーヒーを飲まない（飲めない）。

その結果、ワインは品質と価格が比例していて価格が安定している（価格透明性が高い）。生産者ごとにブランドがあり、品質の良いものを作れば高く売れる。

コーヒーは生産量によって価格が乱高下する「時価」になっていて同じ価格でも品質にバラつきがある。また、生産地（生産国）ごとにまとめて出荷されるので、個々の農家は品質よりも生産量を多くすることに注力する。

参考文献には他にもいろいろと違いが解説されているので是非御一読いただきたい。
特にレストランやホテルでのワインとコーヒーに対する考え方の違いは衝撃的だ。

だから？

ワインもコーヒーも農産物（フルーツ）を加工した飲料ということでは変わらない。
だから「品質に見合った価格のものを飲みましょう。」
というのが LT でお伝えしたかったことのひとつ。

IT 業界はどっち？

ここからが本題。LT したときは前振りに時間かけ過ぎだったか。（反省）

IT 業界の価格決定プロセスはどっち？

あなたの仕事の価値（価格）は誰がどうやって決めているのか？
お客さんは、皆さんの仕事の「何」にお金を払っているのか？
IT 業者は価格に見合った品質を提供できているだろうか？

言うまでもなく、いまの IT 業界は「時価」モデル。
そこで働いているみなさんはコーヒー農家と同じ価格決定プロセスの中にいる。

ワイン農家のように「良いものを作れば高く売れる」ようになりたいと思いませんか？

参考文献：
「コンビニコーヒーは、なぜ高級ホテルより美味いのか」
コーヒーハンター川島良彰 著　ポプラ新書

初出: *2018.10.8* ささみのほん *2*
かいたひと: *khirose*

第 22 話

米もエンジニアも不足している

米が不足してるんですって！

1993 年のことだっただろうか、冷夏による不作で全国的に米不足になり、価格が高騰し購入数が制限されたり、タイなどから大量に米が輸入されたりしたことがある。が、その事を言っているのではない。今まさに米が不足しているというのだ。

エンジニアも不足しているんですって？

エンジニア不足も至る所で取り沙汰されている。特にオリンピックを控えセキュリティエンジニアは深刻な人手不足なんだそうだ。

お前はいったい何を言っているんだ!?

このご時世、よもや報道を鵜呑みにするような人などいないと思うが、米不足もエンジニア不足も嘘ではない。お前がそう思うんならそうなんだろう、

お前ん中ではな

誰がどういう基準で「不足」と言っているのかはちょっとググっていただければ分かるので (ry

また金の話かよ

と思った方ありがとう。前回の「ワインとコーヒーの違い」を思い出していただけただろうか。
一般家庭向けの米が余っているならそれを回せばいいじゃないか。足りないなら値段を上げればいいじゃないか。と思った方は中学生くらいからやり直したほうがいい。モノの値段はそういうことで決まるのではない。特に米とエンジニアは。（だけじゃないけど）
米もワインのように価格と品質が比例している。年毎の豊作不作による価格の変動もよくコントロールされていて、毎年ほぼ同じ値段で同じ美味しさのものを食べることができている。よね？

だから、単に余っているという理由で価格を下げて目的外に販売することはできない。それは市場を破壊して自分の首を絞める。価格に応じた品質が保証されないと価格だけの競争になりあっという間に全体の品質が地に落ちることになる。（コーヒーのようになる。実際はそう単純ではないにしても）
解決策の一つは、外中食産業が「高い米を買って高く売る（原価増を売価に転嫁する）」ことだが、それを消費者（つまりあなた）はそう簡単に受け容れてはくれないだろう。たとえば景気が良くなって皆さんが気前よく牛丼に 500円なり 1000円なりを払ってくれるようになれば、少なくとも今よりはいいバランスを保つことができるだろうが望みは薄い。（牛丼は一例です。他意はありません）

米不足に解決策はない？

簡単に言うと、外中食産業の米不足に解決策はない。わざわざ安い米を生産したがる農家など居ないからだ。ちなみに利益率の改善は解決策にならない。なぜだか適切 (?) な利益率になるように価格のほうが調整されるから。
米全体の生産量・消費量は順調に右肩下がりであることは言うまでもない。農家は生き残りをかけて単価の高い商品にシフトしてくるのでますます米不足は進むと思われる。
だから外中食ではパンか麺を食って自宅では米を炊いて食え。なんて言える訳がない。

エンジニアも同じ

これを読んでいる方はエンジニアが殆どだと思われるので説明は要らないだろう。不足しているのは「優秀なエンジニア」であって、エンジニアは余っていると言っていい。こちらも改善の見込みがない。まぁ、米よりは多少期待できるかもしれないが。

顧客が本当に必要だったもの

ここで言う「優秀なエンジニア」とは、高く買ってもらえるエンジニアのことだ。つまりお金を払う人から見て優秀に見えることを基準にしている。エンジニアはエンジニアリングだけやってればいい。そのとおりかもしれない。だがお金を払う人が求めている「エンジニアリング」にはありとあらゆるエンジニアリングが含まれていることに気がつきで、『特定分野でだけエンジニアリングができます』というのは、エンジニアリ
ひよっ子だ。

続きは本編にて！

第 23 話

どんなに速くなっても遅い

情報の伝達と媒体 (メディア)

一般的に、何らかの情報を伝達するときは媒体 (メディア) が使われます。例えば音声で情報を伝えるときは空気が声の振動を媒介しています。テレビや携帯電話は最近では情報をデジタル化して電波に載せています。光も電波 (電磁波) の一種ですが、光は空気のあるところでは減衰が激しくて長い距離は届きません。それでも光ファイバ (と中継装置) を使うことで数万キロという距離でも高速にデータを伝送できるようになってきました。

伝送速度と遅延 (レイテンシ)

技術の進歩によって伝送速度はどんどん高速広帯域になってきています。光ファイバでは毎秒 10P(ペタ)bps の伝送実験にも成功していて、電波を使う携帯電話やスマホでも、次世代の「5G」ネットワークでは移動体ながら 10Gbps を超える速度で通信できるようになるそうです。

しかし、単に速度 (スループット) が速くなるだけでは解決できない問題もあります。その一つが「遅延 (latency(レイテンシ))」です。

テレビがアナログ放送からデジタル放送になって、時報が無くなったことを覚えていらっしゃるでしょうか?

送っている情報量はデジタル放送のほうが圧倒的に多いのですが、デジタル放送ではある程度のデータを溜めてから符号化 (圧縮) して送り、受け取ったらそれを復号 (展開) してから表示しているので、圧縮展開にかかる時間だけ「遅れて」見えているのです。しかもテレビの性能によってその展開にかかる時間がまちまちなので、すべての視聴者に同じタイミングで時報を届けることができません。

ちなみに、衛星放送では 36,000km 上空の静止衛星を往復する時間 (約 0.5 秒) がかかります。衛星放送がアナログだった頃は 1 秒ほど早く時報を流すことで、だいたい正しい時間に、かつすべての視聴者にほぼ同じタイミングで時報が流れていたのですが、これもデジタル化で廃止されました。

遅延の影響

数ミリ秒ごとにレート (価格) が変化する金融市場では、1 取引あたりが僅かな利益でも、膨大な回数積み重ねられれば大きな利益になります。また、複数市場を使った裁定取引では、値動きにつながる情報をいち早く掴んで他人に先んじて注文を出すことで低リスクでより大きな利益を得ることができます。これら高頻度取引 (HFT) や高速取引 (HST) と言われる手法は、現在では取引の半分を超えると言われています。

ミリ秒を争うこれらの取引では、システムの処理時間を短くすることで利益を最大化する努力は当然に行われていますが、通信の遅延については物理的制約があり難しいところです。対策の一つとして、取引所と同じデータセンターに自社のシステムを置き、構内接続を使って通信の遅延を短くする方式が選択できるようになっていて、HFT/HST　はもとよりアルゴリズムトレードなどシステムトレードでは当たり前に使われています。当たり前に使われているということは、それだけでは競争に勝てないということです。

では、他に通信の遅延を克服する方法はないのでしょうか？

ディープラーニングなど AI による価格変動の予測により、実質的に遅延をマイナスにしようという試みもありますが、これは 100％当たるということはないので、それだけに頼るわけにはいきません。

さて、ここに画期的な遅延解消方法があります！

「量子テレポーテーション」で超光速通信!?

量子もつれの状態にある電子のペアは、一方の状態を観測するともう一方の状態も瞬時に確定します。そしてこの 2 つ粒子のもつれの関係は、どんなに距離を離しても変わりません。何光年離れていても片方が確定すれば瞬時にもう一方も確定するのです。これを利用すれば、光速を超える速度で情報を伝達できる。という話、どっかで聞いたことありませんか？

たいへん夢のある話ですが、実際にはランダムなものを観測しているだけなので意味のある情報は伝達できないというオチです。

でもまだ希望はあります。

フラッシュ・ボーイズという本が話題になったことがあります。(ノンフィクションで、これをもとにした「ハミングバード・プロジェクト～0.001 秒の男たち～」という映画が 2019.9.27に公開されるそうです)

以前は、ニューヨークとシカゴの 2 つの市場の間の通信は 17 ミリ秒ほどかかっていました。それをできる限り最短距離になるように光ファイバを敷設して 13 ミリ秒にしたという話です。最近ではさらにこれを 8.5 ミリ秒にしたという話があります。光ファイバではなくマイクロ波回線でニューヨーク・シカゴ間を結んだそうです。光ファイバをどんなに最短距離で敷設したとしても完全に直線にはできないので余分な長さがあります。
た、光の速度は真空では秒速 30万キロメートルですが、光ファイバを通るときはそ
程度に下がります。無線にするとこれらの制限がほぼ無くなるのでこういうネ
を作ったようです。そして実際に理論上の最短値に近づいたというわけです
※中空光ファイバを使えばガラス繊維を通ることによる速度の低下は改
空といっても真空ではないので減衰が大きく長距離には向かないと

こんなネットワークを日本にも引けるでしょうか？

ニューヨークはあまりにも遠いのでちょっとだけ近い
とを考えてみましょう。実際にそういうネットワーク
出来たら、ロンドン市場の値動きを見ながら東京
きるでしょう。

ロンドン・東京間を直線で結ぼうとする

続きは本編にて！

第 24 話

エンジニアと目の健康（改訂版）

かいたひと **silpheed**
Twitter：@silpheed_kt
Web：http://neo.saitama.jp/

「エンジニア 35 歳定年説」と言われて久しいですが、いつの間にかそれを過ぎた年になってもエンジニアとして生活しています。年齢を重ねるにつれ、徹夜等で無理がきかなくなったりと体の衰えは感じていましたが、先日、糖尿病由来の目の病気を患い、通院開始から約 3 ヶ月で片目が殆ど見えなくなってしまいました。

ことさらエンジニアにとっては、業務の遂行もそうですが技術や業界の動向にキャッチアップし、また継続した学習に目は非常に大きな役割を果たします。多くの情報は視覚情報によって提供されるためです。

今回は「増殖糖尿病網膜症」について、私の事例を紹介します。

末期に急速に進行する「増殖糖尿病網膜症」

「増殖糖尿病網膜症」は、糖尿病網膜症としては末期にあたり、網膜が障害を受け、視力が低下する病気です（参考 1)。網膜はカメラのフィルムやデジカメのセンサーに例えられます。

近視、遠視、乱視といった症状は、屈折異常と呼ばれ、眼鏡やコンタクトレンズで矯正し視力を回復します（カメラでいうレンズを調整します）が、網膜症はセンサーに問題があるので、眼鏡等で回復することはできません。

また、網膜は再生能力がとても低く失った視力は回復しません。

末期になるまで自覚症状がわかり辛く、気がついたときには進行してしまっていることが多いため、糖尿病の治療とともに予防と早期発見が大切な病気なのですが、罹患した場合

- 進行を抑える（網膜の障害を抑える）
- 失った視力を代替手段で補う

ことが必要となります。

糖尿病の内科外来治療

原因は糖尿病のため、内科的には糖尿病の治療を行います。血液検査にヘモグロビンA1c（HbA1c）という項目があり、この数値が 6.2 以上は糖尿病であるといわれます。

HbA1c の値は健康診断の血液検査にあり、また、内科で糖尿病の検査をしてほしいと言えば、遅くとも 2〜3 日で結果が分かります。

食事療法、投薬等により治療を行います。

食事療法は炭水化物、糖類を控える（ただしゼロは NG）ことが目標です。単にご飯やパンや麺類の量を減らし、その他を増やせば OK です。

理想的な食事とは言えませんが、コンビニやスーパーでパック入りのサラダ、インスタント味噌汁やスープ、小さい豆腐のパックを買ってきて食べるだけでも達成できます。血圧の問題もあるため塩分やカロリーにも注意が必要ですが、まず炭水化物の量を減らし、それ以外を多く摂る食事に慣れ何より「継続する」ことが重要です。

投薬は内科の処方薬を飲みますが、場合によってはよく知られたインスリンの投与が必要かも知れません。私の場合は幸いにも錠剤を飲むだけで済んでいます。

血液検査を月 1 回行うため、月 1〜2 回の頻度で通院します。費用は私の場合 1 回あたり 3,000〜4,000 円です。これらの治療により、HbA1c を 8.9 から 6.4（当時）まで下げることができました。現在は 5.3 程度で落ち着いています。

また有酸素運動も効果的と言われています。私は帰宅時に毎日 30 分程度歩くことを続けています。効果があるかは正直分かりませんがしないよりした方が良い、くらいの気持ちです。

眼科での検査と外来治療

並行して眼科で糖尿病がある旨を伝え、一般的な検査（視力測定・眼圧・眼底検査等）を受けます。自覚症状は分かりづらいのですが、飛蚊症（何かが飛んでいるように見える）や出血による黒点、まただんだん見えづらくなっている、といったものも症状です。

私の場合は片目で見たときにものがゆがんで見えることに気づき、異常を認識しました。この状態は「増殖糖尿病網膜症」へと既に進行していることを後に知りました。

糖尿病網膜症は末期になると新生血管と呼ばれる毛細血管が急激に増殖し膜を形成し
す（ゆえに「増殖糖尿病網膜症」と言います）。膜により視野が覆われたり、網膜
張ってしまい網膜剥離を起こすため、レーザーにより新生血管を凝固させる治
す（光凝固）。

光凝固は麻酔の目薬を点眼し特殊なコンタクトレンズを着用の後、専
師がレーザーを目に照射します。

個人差がありますが麻酔のため痛みは少ないのが一般的です
かったですが、レーザーの出力を調整する等で医師が対応し

費用は高額医療となるため限度額適用認定証の区分に
かりました。

光凝固による治療は新生血管の確認のために事
に開始することはできずその間にも病状は進
るかが最重要となります。

続きは本編にて！

第 25 話

引き継ぎは何故うまくいかないのか（改訂版）

かいたひと **silpheed**
Twitter：@silpheed_kt
Web：http://neo.saitama.jp/

人事異動や方針転換等、様々な理由があると思いますが、業務の「引き継ぎ」－自分の業務を他に渡したり、他から渡されたりという場面は案外多いものです。業務を丸ごと渡さずとも、ちょっと他の人のソースを保守したり、自分のそれを頼んだり等であれば、誰しもが経験しているのではないでしょうか。

そしてそれは、うまく行えているでしょうか。

もしかしたら、多少なりとも、予期せぬ不具合の対応等で、何かしっくり来ないなあ、これはエンジニアリングなのだろうか、といった経験があったでしょうか。

本稿では、この「引き継ぎ」（冒頭では便宜上こう書きます）がうまくいかないことがあるのは、あるいは問題だらけなのは何故なのか？　というテーマで、私の業務経験を元に書いてみたいと思います。

ドキュメント / 「足りない」「ない」と / 言われている

私の「引き継ぎ」は、既存システムの運営・保守・改修を、チーム単位（企画、運営、エンジニア、そしてそれらのリーダーも）でほぼ丸ごと業務を受け渡す形で、私はエンジニアのリーダーとして受け取る側の立場となりました。

当初、引き継ぎ元の現場へ行き、業務を行いながら引き継ぐという話でしたが、私は身体上の都合（既刊「ささみのほん」で書いています）で、遠隔での勤務を基本とし、必要があれば現場へ赴くという調整をしてもらいました（最終的には、引き継ぎ先の現場（元とは異なる場所）での勤務となりました）。

機材等の手配に少し時間がかかるということで、現存するドキュメント類をまず調査させてもらうことにしましたが、その際に言われたのが、ドキュメントが「足りない」またはそもそも「ない」ため、基本的に調査ベースの作業となる、ということでした。

全てのドキュメント（殆どが英語）を精査できていない、という話もあったため、実際に全てに目を通してみたのですが、それらは概要は示していても詳細までは分からないものでした。

また、調査ベースというのは、ソースコード、DB、その他周辺ツールの内容を確認し、動作を確認する、という意味のようでした。

手順書の / 項目・意味が / 分からない

諸々の手配が終わり、実際の作業に入りましたが、基本的に、引き継ぎ元からの指示がありそれをこなしていくはずが、引き継ぎ元が定常の業務に忙殺されてしまい、具体的な指示が来ません。そこで、あらかじめ必要になるであろう手順を把握するために、前述のドキュメントを改めて確認していきました。

しかし、例えば、サーバへのデプロイの手順書や、項目のチェックリスト等を見ても、記載項目の意味が分かりません。

- server1にログインし、tool1を起動し、コマンドの 10 番を実行する

とあれば、

1. server1へのログイン方法が不明（アカウント、パスワード、IP アドレス等）。そもそも自分のアカウントがあるのかすら分からない。
2. tool1の起動方法が不明（実行可能ファイルか、何らかのスクリプトか、等々）。

といった具合です。

まず、アカウントの有無を確認したところ、未作成なので申請を行うとのことでした（手配終わってないじゃないか）。

サーバの名前と IP アドレスの対は、サーバサイドのプログラム内の設定ファイルに記載されているとのことで、確かに「server1」でソースコード全体を検索すると見つけることができました。

tool1は、そのまま起動するのでもなく、シェルスクリプトや PHP 、Python 等でもなく、「別のシェルスクリプトのオプションに tool1という文字列を指定する」ことで起動するものでした。これは、そのツール名でドキュメント類を全文検索すると見つけることができました。

ちなみに、それらの、検索で見つけたドキュメントへの参照は、手順書には一切書かれていません。確かにドキュメントが「足りない」のでしょうが、引き継ぎ元はこれを元に作業を行っているはずです。

つまり彼らにとっては「足りている」ことになります。

引き継ぎ元が「足りない」と言っていて、しかし、彼らが「足りている」ものは、引き継ぎ先にとっては「足りない」のです。

続きは本編にて！

第 26 話

VST の負荷分散に関する考察

かいたひと：silpheed
Twitter：@silpheed_kt

はじめに

IT 分野における負荷分散というと、ネットワーク負荷（アクセス増大）対策、分散コンピューティング等が思い浮かぶことが多いと思いますが、DTM の世界でも、PC で様々な処理を行うようになるにつれ、より複雑な処理をこなすための一つの方法として、負荷分散が存在します。

本稿では、その中でも、ネットワークを用いた負荷分散について書いていきます。

DTM 特有の用語が多く登場しますが、基本、DSP（信号処理）をどうするかという観点のため、その他の用語に置き換えて見ていただけると分かりやすいかと思います。

DTM の負荷分散は音切れ対策

DTM で曲の制作中にリアルタイム処理が追いつかなくなるといわゆる「音切れ」が発生します。

音切れの原因で多いのは、CPU のパワー不足と、ディスク I/O の帯域不足です。前者はいわゆるプラグインによる DSP の処理が間に合わない場合、後者はディスクから大量の音声トラックをストリーミング再生する場合（音声トラックによるステムを大量に同時再生する等）に発生します。

この他に、サンプラーで大量の音声データをメモリに蓄えたときに、OS によるスワップが発生して全体のパフォーマンスに影響を与えることがあるかも知れません。

対策 1：フリーズで演算量を減らす

これらを解決するための一般的な方法として、トラックのフリーズが挙げられます。

フリーズは、再生する音声を確定させ（演算済みの音声にレンダリングし、演算を不要とする）、場合によってはプラグインをアンロードすることで、諸々のリソース消費量を削減します。

ただし、フリーズ・フリーズ解除にそれなりの時間がかかること、フリーズ解除しないとそのトラックを編集できないこと等、制作時の快適さを損ないます。

対策 2：ステムの統合でディスク I/O を減らす

ステム（楽器の種類等である程度まとめてオーディオに書き出したデータ）が多くストリーミング再生時にディスク I/O が逼迫している場合、ステムを統合し、数を減らすことが有効ですが、統合したトラックを元に戻すには（例えば、ドラムとギターのトラックを統合したものを、また各々に分割する等）、統合前のプロジェクト（データ）を再度読み込むしかありません。

プロジェクトの読み込みはその規模に応じて分単位での時間がかかることもあり、これも制作時の快適さを損ないます。

対策 3：プロジェクト全体で軽量化できる部分を対応する

プロジェクト自体の構造を見直し、軽量化できる部分を改善するという方法もあります。

例えば、同じエフェクトを複数のトラックでインサーション（後段に直接接続し、特定のパートに対して効果をかける）として使用している場合、それらを一つにまとめる、等が挙げられます。

対策 4：使用可能なリソースの上限を上げる

一番シンプルな方法として、PC 自体のスペックを上げることも考えられますが、これには当然、コストがかかります。

特にエフェクトの処理負荷が高い場合、外部 DSP に処理を任せるという方法もありますが、PC のスペックアップと同様にコストがかかり、また、使用する DSP に対応したエフェクトのみが対象となります。

また、PC の変更や外部 DSP の導入は、制作環境それ自体に大きな変更が入るため、その作業のための人的・時間的コストも考慮しないといけないかも知れません。

対策 5：演算処理を別 PC で行う

一方、プラグインを DAW とは別の PC で動作させ、DAW から制御データ（および処理対象のオーディオデータ）を送り、プラグインによる演算結果を返すという方法があります。

DSP に処理を逃がす方法と似ていますが、PC で動作するプラグインであれば
ること、DAW の入った PC の構成を大きく変更しないで済むこと等のメリ
ます。

どの方法にも一長一短あり、何が最適かは制作シーンにおいて何を
になるでしょうが、ここでは、

- 汎用性（プラグインが DSP に依存しない）
- 金銭コスト（PC のスペックアップや DSP 導入コ
  にあれば有利）
- 環境移行コスト（DAW 導入 PC の環境移行

を重視することとし、この方法について考

続きは本編にて！

第 27 話

サーバレスは誰にとっての弾丸なのか

かいたひと：silpheed
Twitter：@silpheed_kt

はじめに

「サーバレス」という言葉はバズワードとしてはあまり目にしなくなったという印象もあります。最近は「クラウドアーキテクチャ」というのかも知れません。

先日、ようやく「サーバレス」に触れる機会がありました。

ここでは、全くの初心者が「サーバレス」を導入・運用するにあたっての諸々について書いていこうと思います。

製品・サービスは、当時の状況のため、現状とは異なる場合もあります。

また、初心者が短期間のラーニングで対応した内容のため、不適切な理解や認識、設計、実装等あるかと思いますが、「そんなことがあったんだ」的な読み物として、暖かい目で見ていただければ幸いです。

おおまかな要件の整理とサービスの選定

案件の要件は以下の通りでした。

- ソーシャルゲームの開発
- プラットフォームは Android と iOS
- 短納期
- IP 物

要件としては、本当にこれだけしかありませんでした。
とにかく短納期ということで、まず、以下の要件を追加しました。

- クライアントは Unity
- サーバ、API は「サーバレスの何か」
- 「サーバレスの何か」に Unity SDK があること

「サーバレスの何か」としては NCMB の利用実績がありましたが、以前、SDK がまともに動作せず、全て REST API で書き直した経験があり、今回は不採用としました。

また、「次の案件で、クライアント要望で AWS を利用したい」との話があったため、

- ラーニングも行い、次の案件に活かす

ために AWS を採用しました。短納期かつラーニング込みです。
ゲーム部分の詳細も一切ありませんでしたが、一般的なソーシャルゲームに求められる要件から推測し、以下を想定しました。

- ユーザー認証
- リソース管理
- IAP （アプリ内課金）

これらは全て IAP と関連する（IAP の実装は暗黙的に必須）のですが、

- ゲームを遊ぶにはリソース（スタミナ等）が必要
- リソースは時間経過もしくは追加料金の支払い（課金）で回復
- 課金をするにはユーザー毎のデータ管理が必要なため認証が必要

という想定です。
これらを実現するために、以下のサービスを利用することとしました。

Cognito UserPool
ユーザー認証（登録、ログイン、トークン払い出し）を UserPool で実現しました。

SNS （Simple Notification Service ）
SNS に認証済みメールアドレスを設定し、Cognitoに設定する必要がありました（メール送信を行わなくとも）。これは私が誤認しているかも知れません。

API Gateway
Lambda （node.js ）
DynamoDB
API 実行に際し Cognitoで認証させるために API Gateway を使用しました。
Lambda で標準で使える言語のうち実績があったものが node.jsだったため、消去法で node.jsを選択しました。
DB は Aurora Serverless も考えましたが、それほど複雑なデータ管理は必要なさそうだったため、KVS 利用の実績を作るために DynamoDB を選択しました。

S3
CloudFront
アプリがダウンロードするリソースを置くストレージとして S3 を選択
CloudFront の利用については速度面、コスト面での判断が本来は
とりあえずの選択として利用することとしました。

CloudWatch Logs
主に API Gateway と Lambda 関数のログを記録す
択しました。後述しますが、DynamoDB へのログ

続きは本編にて！

第 28 話

とある大手町 NOC の遍歴

かいたひと：Aki@nekoruri (サークル「めもおきば」)
書いた人の日常：メイドさんかわいい

はじめに

この記事は、大手町から最も近いとされる自宅ラッカーの一般的な歴史を淡々と描く物です。過度な期待はしないでください。

自宅ラック

もちろんご存知だとは思いますが、「自宅ラック」とは、サーバラック（狭義には IEC 60297-3 シリーズにて規定される 19 インチラックのうち、音響機器用ではない奥行きが比較的長いもの）を、自宅に置いている場合に、そのラックのことを指します。

自宅ラックがなぜ必要かと言われれば、そこに 19 インチ規格でマウントできる機器があるからです。いっぱんてきなエンジニアであれば、業務で試してみたいと思う技術について、まずはごかてい で遊んでいることでしょう。普段から遊んでいない技術を業務で使うなんてとんでもない！

であれば、サーバであったりネットワーク機器であったり自然と 19 インチ規格の機器が集まっているはずで、当然自宅にサーバラックがあるのは自明です。

冗談はさておき、PC やサーバ・ネットワーク機器が何台もあるのであれば、丈夫な床がある環境ならキャスタータイプのサーバラックが実際にオススメです。ラックごと手前にちょっと引き出して背面の作業ができますし、部屋のレイアウト変更もそのまま移動ができます。中古で探すと 2 万円ぐらいであるようです。

そんな建前のもと、ラックを自宅や心の中に持ってしまった人のことを「自宅ラッカー」と呼ぶわけです。

前史：とある大手町 NOC の誕生

人類は 1999 年 7 の月を乗り越え 2000 年問題をつつがなく過ごし、まともに動く初めての一般ユーザ向け Windows である Windows 2000 が登場した西暦 2000 年、とある大手町 NOC は生まれました。まだ 19 インチラックはなく、ご家庭に一般的な 128kbps 専用線である OCN エコノミーがただ L アングルで組まれた棚に収容されていました。当時はまだ IPv4 アドレスが余っていたので、なんと/28 のネットワークにインクジェットプリンタまでグローバルアドレスを付けて大学から自宅に印刷できる夢のユビキタス環境

でした。

その直後、街頭で ADSL モデム入りダンボールケースが無料配布したことで一躍有名になった Yahoo!BB や、IPv6 実験目的で e-accessの ADSL 併用となり賑やかなネットワーク構成が拡がります。当時はソース IP アドレスの egress filterもなく、OCN の固定 IP アドレスをソースにしたパケットを ADSL 経由でインターネットに流すことができ、上りの方が 10倍以上速いと喜んでいたものです。

図 28.1: 懐かしの MN128 と ADSL モデム群

古代：19 インチラックの文明開花

そこから 4 年たった 2004年の正月休み、ヤフオクで少々小ぶりながらも自宅のサイズぴったりの 19インチラックを発見し、ようやく自宅ラッカーの仲間入りをします。幸いに車に入るサイズだったので出品者の倉庫まで取りに行ったのが懐かしいです。せっかく 19インチラックなのだからと直後に PANDUIT のパッチパネルまで買っています。

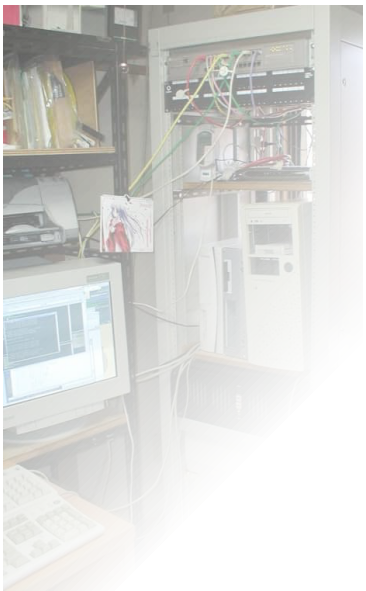


図 28.2: 会いに行けるフルラック

この年、ラック導入記念と言うことで USEN
エコノミーから B フレッツの OCN IP8
サービスで、PPPoE しない素のネット

続きは本編にて！

第 29 話

寄付の自動運用について

かいたひと：Aki@nekoruri (サークル「めもおきば」)
秋葉原生まれ大手町育ちの歌って踊れる江戸っ子フルスタッククラウドエンジニア。0 と 1 が紡ぐ「ゆるやかなつながり」に魅せられ早 20 年、SNS と CGM の力で世界を幸福にするのがライフワーク。最近は IoT ビジネスをメインに、技術系同人誌を書いたり、セキュリティ・キャンプや SecHack365 で飛び回ったりする日々。

市民、幸福は義務です。
あなたは幸福ですか？

ウガンダの恵まれない子供たちに援助を！

みなさん Vim つかってますよね？
Vim は素晴らしいエディタです。カーソル操作や文字列入力を含めて全ての操作をコマンドとみなすことで、圧倒的な作業効率をエンジニアにもたらしてくれます。

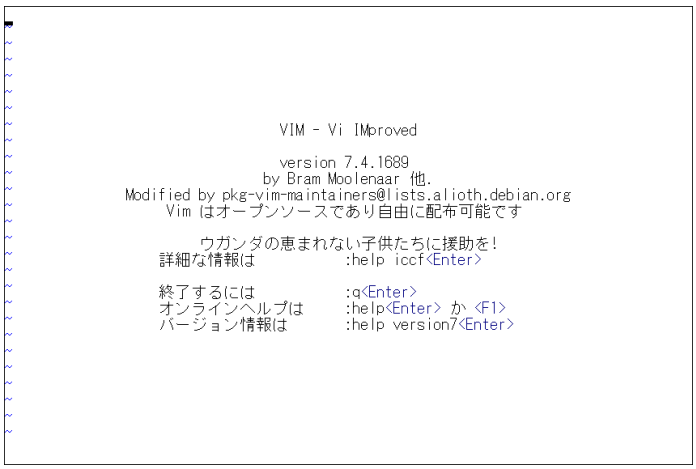


図 29.1: Vim

さて、ご存知の通り、Vim はチャリティウェアです。ライセンスとしては GPL 互換ですが、ウガンダの孤児達への寄付が奨励されています。

リスト 29.1: runtime/doc/uganda.txt

```
Vim is Charityware.  You can use and copy it as much as you like, but you are
encouraged to make a donation for needy children in Uganda.  Please see |kcc|
below or visit the ICCF web site, available at these URLs:

        http://iccf-holland.org/
        http://www.vim.org/iccf/
        http://www.iccf.nl/

You can also sponsor the development of Vim.  Vim sponsors can vote for
features.  See |sponsor|.  The money goes to Uganda anyway.

The Open Publication License applies to the Vim documentation, see
|manual-copyright|.
```

自動化だいじとてもだいじ

ただ単に一度寄付するだけであれば、上記の iccf-holland.orgにある「Make a donation」ボタンから PayPal に飛んで、そのまま寄付できます。総額 1 万以上寄付していないとはてな匿名ダイアリーで「Vim 使い失格」と煽られてしまう世の中[*1]なので、一度に 1 万円はちょっと大きければ繰り返し寄付しても良いでしょう。

ですが、我々はなんだ！ そうだ！ エンジニアだ！

エンジニアならば、繰り返し行う作業は自動化するべきです。

寄付の自動化

というわけで、毎月月額寄付を設定してみましょう。

ICCF のサイト（iccf-holland.org）を開いて、Make a donationボタンをすぐに押さず、寄付の詳細ページを開きます。

続きは本編にて！

[*1] https://anond.hatelabo.jp/20160211231359

第 30 話

Minecraft で整地する話、あるいは Docker でインフラを整備する話

かいたひと：Aki(秋葉原生まれ大手町育ちの歌って踊れる江戸っ子インフラエンジニア)
Twitter：@nekoruri

はじめに

みなさん、整地していますか？
本章では、Minecraft のサーバインフラを Docker で管理する話をします。

背景

平地をならした俺たち整地部隊は、
Windows Update を掛けられサーバが停止されたが、
デスクトップ PC を脱出し VPS に潜った。
しかし、レガシーでくすぶっている様な俺たちじゃあない。
運用さえできれば暇次第で何でもやってのける命知らず。
チャンクを y=4 にし、巨大なバイオームを粉砕する。

俺たち、整地やろう ssmjp チーム！

設計方針

Minecraft のサーバ運用にあたって、以下の設計方針を立てました。

- 基本的に自分でスクリプトを書かない
  - 実績のある既存のツールを活用する
- きちんとバックアップする
  - 当然リストアできること
  - リストアがしやすいこと
- きちんと監視する
  - とりあえず Makerel の無料プランで良いかな

構築手順

それでは作っていきます。

環境整備

まずメモリが 2G か 4G のサーバをどこからか用意します。今回は、あざとかわいいこのはちゃんで有名な ConoHa VPS を利用します。公式 Minecraft テンプレート？ 知らない子ですね。

OS としては Ubuntu 18.04を選びました。まあどうせ Dockerの中しか触らないので、Dockerが対応している distroなら割とどれでも良いです。今回は、コンテナのオーケストレーションに docker-composeを使うので、Docker Community Edition[*1]に加えて、docker-compose[*2]を導入します。

```
$ sudo apt-get update
$ sudo apt-get install \
      apt-transport-https \
      ca-certificates \
      curl \
      gnupg-agent \
      software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo apt-key fingerprint 0EBFCD88
$ sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io

$ sudo curl -L \
  "https://github.com/docker/compose/releases/download/1.24.0/docker-compose-$(uname
  -o /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
```

簡単ですね。docs.docker.comにあるコマンド垂れ流しで OK です。ちなみに第
最後の chmod +x を忘れて、時間を捨てました。

バックアップ先の準備

続いて、バックアップ先に利用する Amazon S3側の準備をし
二つ、実際にデータを保存する S3 Bucket と、そこにファイル
User/IAM Policy です。

今回は AWS 外部の ConoHa から送信するため IA
スキーが必要になりますが、たとえば EC2 上か

続きは本編にて！

[*1] https://docs.docker.com/install/linux/dock
[*2] https://docs.docker.com/compose/inst

第 31 話

Minecraft で整地 (略) する話 (2)

かいたひと：Aki(秋葉原生まれ大手町育ちの歌って踊れる江戸っ子インフラエンジニア)
Twitter：@nekoruri

はじめに

みなさん、整地していますか？

本章では、Minecraft で整地をする話、あるいは Docker でサーバインフラを運用する話をします。

前回のあらすじ (きちんと運用 #とは)

前回の記事（「ささみのほん 3」を読んでね！）のとおり、Docker を使った Minecraft サーバの運用を開始しましたが、いくつかの問題が発生しました。

サーバを落とすとバージョンが上がる

今回使っているコンテナイメージ `itzg/minecraft-server` は、一般的なコンテナ化のようにソフトウェア全体を含むのでは無く、コンテナ起動時に公式サイトから Java パッケージをダウンロードしてきて実行します。コンテナイメージを再ビルドせずに環境変数のみでサーバのバージョンや設定を変更できる、いわばランチャーの形態を取っていますが、設定が不十分な「デフォルト」状態のままだと意図しない動作が発生します。

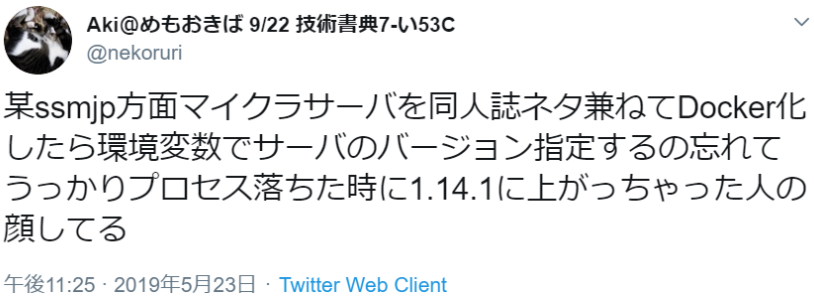


図: https://twitter.com/nekoruri/status/1131566781837873153

Minecraft では、深遠な理由によってサーバプロセスが死ぬことがまれに良くありますが、環境変数でバージョンを固定しないと、うっかりメジャーバージョンが上がってしまったりします。

というわけで、きちんと指定しましょう。

docker-compose.yml

```
mc:
    image: itzg/minecraft-server
    ports:
        - 25565:25565
    expose:
        - '25575'
    volumes:
        - mc:/data
    environment:
        EULA: "TRUE"
        MAX_MEMORY: 1G
        ENABLE_RCON: "true"
        VERSION: "1.14.4"
    env_file:
        - minecraft.env
    restart: always
```

また、環境変数を設定したはずなのにバージョンアップしてしまう事象がありました。これは docker-composeのコマンドの動作を理解していなかったためで、restart コマンド[*1]では設定ファイルへの更新は反映されず、upコマンドを使う必要がありました。

```
docker-compose restart

Restarts all stopped and running services.

If you make changes to your docker-compose.yml configuration these changes are not reflected after running this command.
```

バックアップが止まっていた

一般的に一番やばいやつです。

restart: always をつけましょう。明示的に止めない限り、コンテナのプロセスが終了すると再起動してくれるようになります。

docker-compose.yml

```
backup:
    build: https://github.com/nekoruri/docker-mc-backup.git
    volumes:
        - mc:/data:ro
        - backups:/backups
    links:
        - mc
    environment:
        RCON_HOST: mc
    env_file:
        - minecraft.env
    restart: always
```

続きは本編にて！

[*1] https://docs.docker.com/compose/refer

第 32 話

夫婦喧嘩からはじまる CI/CD

とある会社に勤めるエンジニア夫婦のおはなしです。
こういう夫婦がどこかにいたのかもしれない的な。

やまさき

それはとある日の夕食後、夫婦の雑談から

夕食を終え、日本酒とちょっとしたものをつまみつつ、動画配信サービスで動画を見ながら血小板ちゃんかわいいとか言っている夫に対し、あーはいはい、と流す妻。

いつもどおりのいつもの暮らし。
そんな中、ふと口をついて出た言葉。

問題提起

妻「うちの会社でも CI/CD をしたらいいのにね」

いつもどおりの夫婦の会話。
何気なく、無邪気に発せられる疑問から始まる夫婦の対話。

我が家は夫婦ともにエンジニアである。
夫はアプリケーション開発者、妻はインフラエンジニアで同じ会社で働き、夫婦となった。
我が社でもシステム開発の内製化が始まり、それまではほぼ外部ベンダーにシステム開発を依存していたのを開発チームを立ち上げて自分たちで開発するようになってから 1 年ほど過ぎたあたりのことだった。

内製化立ち上げ当初は開発者の人数も片手で数えられるぐらいだった。
はじめに 1 つのサービスを作り上げ、それから徐々に既存社員で前職等で開発経験がある者の異動や中途採用による即戦力となる開発要員の増強を行い、開発に興味のある社員と社内 SE 枠採用の新卒社員をイチから教育していった。
内製化に舵を切ってから 1 年が経つ頃にはようやく開発者が 20 名を超えたぐらいになり、開発・運用しているシステム数も 10 を超えてきていた。

即座に却下（涙

夫「え……うちの会社には要らなくね？」

夫、妻、それぞれの状況

夫が所属する開発チームは立ち上がってから 1 年、まだまだ開発要員は足りない状況であるものの社内からの新規システム開発や稼働中システムに対する要望への対応で手一杯で、とてもサーバーレス構成やマイクロサービス化を検討・検証するほどの余裕がなかった。

データベースやロードバランサーは PaaS を使用しているものの、Linux の仮想サーバー上でアプリケーションは Apache と PHP を使って動かし、データベースは MySQL を使用する構成である。

そして、開発で使用しているリポジトリも Subversionで、デプロイ作業はリーダークラスの社員が手動で行うという、ややレガシーで暖かみのある開発・リリース体制になっていた。

事業会社でシステムの内製化をする場合、いかに開発者を集められるかがカギである。しかし、中途採用はなかなか応募が来ない。求人を出してから半年ぐらい経ってやっと 1〜2 名が入社してくる程度である。

新卒入社の社員は会社での社内 SE 枠採用で 10 名程入ってくるが、文系理系問わずで採用しているのでプログラミングは全く初めてという者はもちろんいる。

理系学部卒でも C 言語を少しだけ……といった者も多く、春から夏にかけては新人への教育で既存社員の手が取られ、開発業務にかけられる時間がどうしても少なくなってしまう。

それに、内製化立ち上げ期なのでまずは開発がちゃんとできる体制がとれるぐらいの開発者を確保することを主眼とし、システム構成や開発手法の選定については後からでも変えられるので「自分たちがいま一番やりやすい方法」としてこの形になっていた。

会社では 2014年からクラウド利用をしており、内製で作ったシステムもクラウ
稼働している。

妻が所属するインフラチームも、物理サーバー担当やネットワーク担当
クラウド担当とチームの中で担当分野が分かれているのだが、他の担当
に対しクラウド担当は妻 1 人だけという時期が数年間あり、その間
既存システムのクラウド移行や新規システム導入や運用をすべて

外部ベンダーに開発を依頼しているアプリケーション
合に既存システムに対する修正箇所が少なく済むリ

具体的には、OS を含むプロダクト環境のクラ
は OS 環境上にアプリケーションを構築する

リフトアップ型のクラウド移行となる

続きは本編にて！

第 33 話

WSL を理解して楽しく使う話

かいたひと：みむら（a.k.a. 親方, 浴衣の人）
Twitter : @mimura1133 / Facebook : mimura1133
Inside IT 主催, CTF for Beginners cofounder.

自己紹介

はじめまして。みむらと申します。浴衣を着て、いろんなひとに IT 技術を気兼ねなく楽しんで活用してもらえる環境作りのため、日々何か出来ないかと奮闘しています。

現在はアプリの脆弱性診断員ですが、かつては Windows のカーネルモードドライバの開発やフォレンジック（マルウェア解析など）を業務で行っていたこともあり、基本的には低レイヤーが興味関心の中心です。

はじめに

2015 年にリリースされ、Windows as a Service と銘打ってアップデートが続けられている Windows 10 ですが、今日までに多くの新機能が搭載されました。マルチデスクトップや Cortana といったものから Windows Hypervisor Platform や Windows Sandbox、ReFS ver 3.x などなど、これまでに様々な機能が追加されました。

今回紹介する WSL (Windows Subsystem for Linux) も、そんな追加された機能の 1 つです。この機能についてどのような仕組みで動作し、どのような機能が提供されるのかについて説明したいと思います。

そもそも WSL とは

WSL は "Windows Subsystem for Linux" という名前の通り、Windows 上で Linux のユーザアプリケーションを動作させる機能です。一般ユーザ向けには Windows 10 Fall Creators Update (1709) 以降で利用可能になりました。

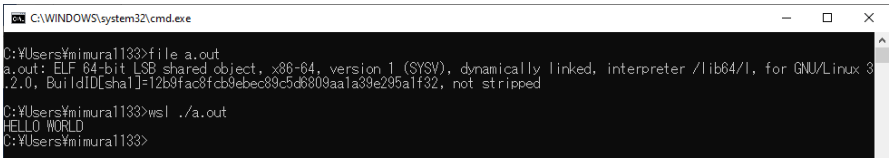


図 33.1: こんな感じで Windows から ELF が実行できる

Windows 環境で Linux の資産を使う方法としては MinGW や Cygwin を利用するというのが一般的でした。ですが MinGW は Windows で用意されていない Linux の機能を利用するプログラムを移植しようとした際に難があり、Cygwin は実行速度において難がある時がありました。またどちらも実行ファイルで提供されたプログラムは利用できないという問題があります。

一方 WSL では Windows 上で 64bit の ELF ファイルを直接起動することが可能となっていることやプログラム側の修正がほとんど不要なまま動くようになっているため、MinGW や Cygwin よりも互換性が高く、より柔軟に Linux の資産を利用することが可能になりました。

また、読者の皆様の中には Windows で提供されていた Subsystem for UNIX based Applications (SUA) という機能を覚えている方もいらっしゃると思います。SUA は POSIX システムとの互換を取るためのものでしたが、WSL は最初から Linux システムを想定して作成されていますので、SUA よりもより柔軟且つ気軽に Linux の資産を活用することが可能になっています。

それでは一緒に WSL を使うための方法について見て行きましょう。

WSL を使うには

WSL をお使いの環境に導入する方法は極めてシンプルです。 Windows Store からお好きな Linux ディストリビューションからお好きなものを選択してインストールするだけです。

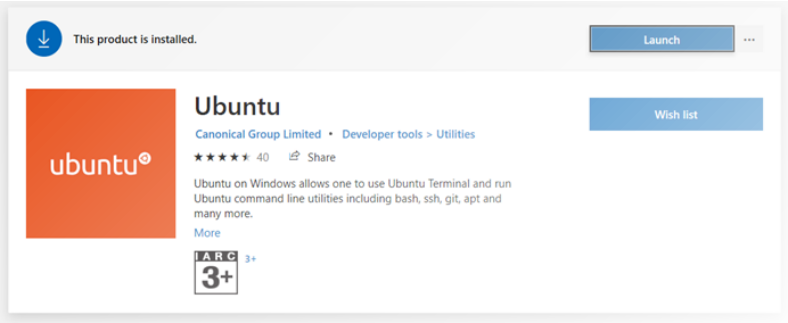


図 33.2: ストアの一例

インストールが完了した後は、スタートメニュー内に「Ubuntu」という
すのでそこをクリックして起動します。

図 33.3: スタートメニュー内

続きは本編にて！

起動した環境内では Linux システムに

第 34 話

槓の光 -初心者からベテランまで、今すぐ実践できる麻雀の考え方、あるいは宗教-

かいたひと：教祖様 a.k.a. pochi
Twitter: @sasakipochi
Facebook: sasakipochi

ssmjp 麻雀部について

ssmjp ではコミュニケーションツールとして Slack を活用している。そこには「#麻雀やろうぜ」という「麻雀したい」と言えば面子がすぐに集まる素敵なチャンネルがあって、ssmjp 麻雀部では日々麻雀を楽しく打つことができている。Slack は良いね。[*1]

ssmjp 麻雀部は、健康マージャンで記録を全部残す、というスタイルで、お財布には優しいんだけど、記録が残ると、負けたときに本当に悔しいんだよね。そして、ずーっと負け続けてる、みたいなのがわかると、さらにさらに悔しい気分になるんだよ。

「槓の光」の誕生

2019 年 6 月上旬、何をやっても勝てないと思うほどツキに見放された時期があって、もうどうとでもなれ、という気分で意味のない槓をしまくってみたんだ。するとあら不思議、4 日連続でトップ。ほんとに、気が狂ったようにわけのわからない槓を沢山してるのにナゼか和がれてしまう。麻雀は「槓」をするゲームだったんだ!! 新しい宗教「槓の光」が誕生しましたよ。

具体的な打ち方

こんな方針で打てば OK。

とりあえず槓を目指す

槓できるときには槓をする、暗槓できるときには暗槓、明槓できるときには迷わず明槓する、という方針で良い。

[*1] #麻雀やろうぜ、は ssmjp の Slack の中で最もアクティブなチャンネルだぞ!!

ポンは 1 鳴きで

ポンできる牌が捨てられたら即ポンをする。そうすれば同じ牌をツモってきたときに槓できる。

対子が 3 つあったらポンをしていく

ポンをしていけば槓ができるはず。シンプルに槓を目指していけば意外となんとかなる。

槓できない牌は捨てて OK

すでに他の人が捨てている牌は、もう槓することができないので、遠慮なく切っていけば良い。何を切れば良いか迷ったときには、そういう考え方で。

役は対々和と混一色を覚えよう

ポンや槓を積極的にしていくと、和がれる役が限られていくけど、対々和と混一色を目指せばだいたいなんとかなる。そしてこの 2 つの役はとても覚えやすい。

字牌は大事にする

対々和と混一色で和がる場合、字牌を持っているメリットはそれなりに大きい。早めに切らないで持っていると良いことがあるかもしれない。

ドラは作るもの

ついついドラは大事にしてしまいがちになるけど、カンをするとドラは増えるので、ドラがいらないときには、自分で作る気持ちで切ると良い。

槓をすることのメリット

実際にこの打ち方にはメリットも多い。

迷いがなくなる

麻雀は、捨てるべきかどうか、鳴くべきかどうか、リーチをかけるべ
うことが多いゲームなんだけど、槓を目指す、という方針で打つと
なるんだよね。槓できない牌は捨てれば良いし、ポンと槓は無条
チは鳴いてるとできないし。

振聴 (フリテン) が減る

捨牌に迷ったりしている初心者は例外なく振
ンツ）から 1 枚欠けた搭子（ターツ）の処

続きは本編にて！

第 35 話

コミケで技術同人誌をゲットしよう 一般参加編

かいたひと: ありすゆう (@Alice_You)
インフラエンジニアの毒舌な妹 (@infra_imouto)

お兄ちゃん、コミケの技術同人誌は、アウトプットじゃない、パッションの奔流だよ。

はじめに

最近は、技術同人誌がブームです。アウトプットを、同人誌という形で行うのが技術同人誌。そして、技術同人誌を頒布するためのイベントも、技術書典をはじめとして、複数行われています。ですが、技術書典などの、技術系の同人誌即売会がまだなかったころ、技術同人誌はなかったのでしょう？

そんなことはありません。実は、技術書典の開催以前から、技術同人というジャンルは存在していました。そのころは、技術同人誌は、どこで、作者から読者へと渡っていたのでしょうか。

それまで、技術同人誌を頒布する数少ない場であったのが、夏と冬に開催されるコミックマーケット、通称コミケです。技術書典などの、いわゆるオンリーイベントが勃興した今でも、コミケが技術同人誌を頒布する場のひとつとしてあり続けています。

コミックマーケットで技術同人誌を求めようという皆さんに、コミケで本を手にするためガイドをさせてもらえばというのが、本稿の目的です。

本稿は、コミケに行ったことがないという人をターゲットとしています。そのため、コミケでの技術同人誌の事情を説明するとともに、コミケに関する情報に触れる内容になっています。もし、本稿を読んでコミックマーケットに興味を持った場合、かならず公式 Web サイトを参照し、発行物のカタログ冒頭の諸注意を確認してください。余裕があれば、カタログ巻末のマンレポ (マンガレポート) ページに書かれた TIPS や生存術を読んでください。

コミケとはどんなイベントなのか

見たこともないものがあるから

まず、技術同人誌を入手する場としてのコミックマーケットについて説明しましょう。技術書典で同人誌即売会のイメージをつくっている人を想定して、コミケというものが何かを説明します。

コミックマーケットとはどんなイベント

コミックマーケットとはどんなイベントなのでしょうか。名前だけは知っている、という人も多いかと重います。そして、おそらく、盆暮れのニュース映像としての人混みの映像を見たという人も多いでしょう。

コミックマーケットは、40 年以上の歴史を数える、総合ジャンルの同人誌即売会で、間違いなく国内最大のものです。開催回数は、2019年の 8 月に開催されたコミックマーケットで、96 回目となります。以降、特定の開催回のコミケについては、慣例に従って、コミケ 96 や、それを略して C96 と記載します。主催は、コミックマーケット準備会という任意団体です。会場との取引の都合で法人化している部分はありますが、原則として、有志が主催し、運営しているイベントです。

総合ジャンルとはなにか

総合ジャンル、というのは、どのような形態を指すのでしょうか。

総合ジャンルというのは、一次創作 (同人サークルオリジナル)、二次創作 (既存作品のファン活動、ファンジン)、評論情報、創作など、コミックや文章という表現の形態をそれ以外も区別せずに受け入れる、ということです。それは、性的表現を含む成人向けであるか、そうでないか、という別の軸でも、区別なく受け入れる、ということでもあります。

そして、それこそが、技術同人というジャンルを内包する下地でもありました。

コミックマーケットは、すべての表現を受け入れる、という絶対的な方針があります。その方針故に、技術同人誌も、多彩な表現のひとつとして早くから受け入れられ、現在にいたります。筆者の知るところで、20 年前のカタログにも、技術同人のサークルの存在を見ることができました。

コミックマーケットの会場

コミックマーケットは、最近の 20 年程の開催において、有明の東京ビッグサイトを全館使用して、会場としています。ただし、2019年から 2020年の間は、東京オリンピックのプレスセンターとして東京ビッグサイトが使用される関係で、変則的な会場使用となります。

本稿で書いていることは 2019年 8 月末時点の公式発表に拠っていますが、変更で
こともあります。最新の情報はコミックマーケット公式の Web ページとカタロ
を参照してください。

コミックマーケットの開催時期

コミックマーケットは、8 月の旧暦のお盆にあたる週末と、1
を含む年末に行われます。前者を夏コミ、後者を冬コミと通
期間は、3 日間が通常とされています。ただし、2019
面積が減ることから、変則的に 4 日間開催となってい
正直なところ、社会人には優しくない日程です
る日程を選択すると、残ったのがお盆と大晦日
2019年の夏コミである C96 は、8 月 9

続きは本編にて！

第 36 話

寓話 「放送システムの運用」

かいたひと：みうみう
Twitter: @miu_crescent

ssmjp

地方住まいの人間が ssmjp という勉強会に参加してみての感想は「なにこれ楽しい」でした。悪ふざけしているのかと思ったら、とっても真面目に運用についてのお話をしているとっても素敵な会。

勝手にシンパシーを感じてしまったので、未経験とか空気わかってないとかも気にせず、ささみ本に飛び込んでみました。これなら地方からでも参加できますし。混ぜていただいて感謝です。

ssmjp 自体には、まだ数回しか参加できていないので、もっと参加したい！　登壇してみたい！　ネタはこれでいいかしら。

運用ってなんだろう

でも、ささみに参加しているなかで１つ気づいてしまったのです。『私が聞いたことある運用と、ささみに出てくる運用は、似て非なる全くの別物なのではないか』と。具体的にどこが違うのか、みんながとっても高次元な話をしているのか。そもそも私の思い過ごしなのか。なんなのか。

そこで、

私が伝え聞いた放送局における運用の話を素直、かつ盛大に脚色しつつ、綴ってみることにしました。この寓話？　をきっかけに議論が生まれ、私の感じているモヤモヤが少しは晴れるのではないか。という淡い期待を込めて。

それでは、張り切ってどうぞ！

マスターさんのとある１日

マスターさんと放送監視

放送局にはマスターさんという人たちがいます。主調整室、通称マスター室というお部屋に住んでいます。24 時間、365 日絶え間なくお仕事をしています。その主なお仕事は〈放送の監視〉です。

コンテンツではなく、信号を監視しています。地デジになったので、信号という言い方は不適切かもしれませんが。CM が一番大切なので、番組本編が始まってからトイレに行

きます。基本的に 2 人以上が常駐してますが、深夜は交代で仮眠をとるので、そのときはトイレは我慢です。ほんとに大変なときは相手を起こします。

放送設備は全く同じ装置が 2 系統ホットスタンバイしています。マスターさんは機器に故障があったらその場で系を切り替えます。SLA という言葉を使うと 100%がマスターさんには求められます。なにがあっても放送を意図した通りに継続します。

他にもマスターさんは中継回線用のアンテナを操作したり、衛星の手配をしたりと大忙しです。マスターさんがあるボタンを押すと「ピロリンッ」という音とともに地震速報や交通情報が出てきます。数少ない目に見えるお仕事です。

これらを放送の監視をしながら、こなしています。

マスターさんと運用自動化

平日の夕方になると、マスター室には放送さんがやってます。

「データチェックしまーす」

放送さんは毎日、翌日分の放送データを作成しシステムに流し込んでいます。それが正しく反映されているのか最終段の端末で確認をしに来るのです。放送さんが作るデータは秒単位です。すべての番組や CM、提供などを正しくプログラムし、ID の紐付けなどが間違っていないかを確認します。その 1 日分のプログラムデータを印刷し、マスターさんに渡すと放送さんの 1 日の仕事が終わります。

マスターさんは放送さんの印刷してくれた紙を伝統の赤鉛筆でチェックしながら、本当にミスがないのかを確認します。人為的なミスはゼロにはできませんし、既知のバグをもう一回踏むわけにはいきません。放送事故は絶対にあってはいけません。謝罪する相手は視聴者のみなさんはもちろん、スポンサーや総務省も含まれます。大問題です。

放送は意図した通り 100%継続され続けなければなりません。そんなことが起きてしまわないように、何重にもチェックを重ねます。このデータチェックが正しく終われば、マスターさんとしてはひと段落です。やっと放送の監視に集中できます。

何も起きなければ、あとはシステムが自動的にやってくれるからです。あとはまったり放送監視です。

そう、なにも起きなければ……

マスターさんと緊急対応

「緊急地震速報を受信しました」

なんの前触れもなく、けたたましいアラーム音とともに、感情の入っ
生されました。マスターさんは立ち上がり、アラームの内容を確認し

「マスターさん！！ 10分後にマスターカット！」

インカム経由で割れた声が聞こえます。報道さんが緊
放送局では災害時に防災・減災のため、緊急放送を行
最も優先順位が高いのは人命に関わる放送です

「沿岸部に、津波警報！」

続きは本編にて！

付録 A

かいたひとたち

じゅんじゅん

ssmjp の運営（受付・広報担当）な人。おもしろい。

togakushi

元主宰。**アウトプットしないのは知的な便秘**という言葉が本人の知らないところで流行ってる人。すごい。

@_keihino / けい

何やってるか自分でもよくわからない人。なんか電子工作もクラウドもできるフリしてる中途半端な人。フルスタックエンジニアのフルってどこから?って思ってたら、最近一級建築士の人が TCP/IP の話ししてたのを見て、マジやべえと思いました。

yakumo3

ssmjp の主宰。呼び捨てにしてもやくもさん。

tigerszk

とある診断員。すごい。

東雲翡陽

いろんなことができるひと。すごい。

Mary

チューバ吹きの料理人。すごい。

khirose

アラフィフな人。すごい。

silpheed

某雑誌の元ライターな人。すごい。音楽もできるしハードもつくれる。すごい。

Aki@nekoruri

フルスタックエンジニアな人。サーバーレスな人。本人のサークルもあるよ！

やまさき

JAWS DAYS 2019 のコアメンバー、ssmjp ではヤマサキ春のサメ祭り発起人の一人。すごい。

みむら

若いはずなのに年齢不詳なひと。だいたい何でも知ってる引き出しの多いひと。すごい。

pochi

バイオリンが弾けるレンタルおじさん。すごい。無職 1 周年。やばい。

ありすゆう

コミケでパッションを奔流するひと。すごい。

みうみう

北の大地からの遠距離執筆参加者。札幌の地に ssmjp を普及してくれてるひと。すごい。

付録 B

初出

ささみのほん 1

- 意識低い勉強会運営の話:改 2
- 今日も残念
- RAID2 は何故使われないのか
- ESP-WROOM-32 用ブレイクアウトボードのステンシル作ってリフローしてみた
- PDCA に対する誤解と真実
- とある大手町 NOC の遍歴
- とある診断員の回顧録~脆弱性診断の現場から愛をこめて~
- エンジニアと目の健康（改訂版）
- 時間のないエンジニアに贈る料理の運用の話

ささみのほん 2

- ワインとコーヒーの違い（ふたたび）
- ライダーベルトを分解してつくりかける話
- Mastodon を運用してみてる話
- “ネットマスク”ってなに？
- 引き継ぎは何故うまくいかないのか
- 寄付の自動運用について

ささみのほん 3

- 米もエンジニアも不足している
- Ansible のプロジェクトドキュメントを Sphinx でイイ感じに管理してみる件
- 料理の運用の話
- VST の負荷分散に関する考察
- おうちインフラ運用事情の話
- PHP でも Raspberry Pi がしたい！
- Minecraft で整地する話、あるいは Docker でインフラを整備する話
- "ネットマスク"ってなに?（つづき）

ささみのほん 4

- どんなに速くなっても遅い
- ささみのほん 3 ができるまで
- 勉強会を運用する話
- 槓の光 -初心者からベテランまで、今すぐ実践できる麻雀の考え方、あるいは宗教-
- 【未解決】Raspbian buster リリースでモノラルアンプがちゃんと使えない
- ひとり NOC をやった話。あるいはインターネット接続を提供するということ (1)
- Minecraft で整地 (略) する話 (2)
- 寓話 「放送システムの運用」
- WSL を理解して楽しく使う話
- 夫婦喧嘩からはじまる CI/CD
- サーバレスは誰にとっての弾丸なのか
- コミケで技術同人誌をゲットしよう 一般参加編
- HTTP ステータスコードの話
- 真のエンジニアになるには

あとがき

ささみのほん 1 の印刷物がなくなってしまい、過去の作品が手に入らなくなったので「電子版で復活させようぜ！！」からはじまったこの企画。

当初は原稿が全部揃うかもあやしく「（あるものだけを集めてよりぬいた）よりぬき」という趣旨でした。

探してみたら原稿は全部集まり、ついでだから誤字脱字を直したり、書き足してアップデートしてみたり。

そんな感じでせっせと作ってる最中にささみのほん 2 の在庫も無くなってしまうありがたい事態に。

＊「こうなったら『ささみのほん 2』の内容も入れちゃおうぜ！」
＊「ぜんぶ載ってるから『よりぬき』じゃなくなったね！」

ｱｯﾊｲ。

『ささみのほん Archives 2019(1〜2)』お楽しみいただけたでしょうか。

謝辞

今回表紙を担当してくれた silpheed さんに感謝！！

ダウンロードカードからダウンロードシステムまでの配布周りを担当してくれたしののめさんに感謝！！

同人部（だけじゃないけど）サーバーの基盤メンテナンスをしてくれているじゅんじゅんさんに感謝！！

この本を手に取り読んでくれた、あなたに感謝！！

あとがきのあとがき

ささみのほん Archives の頒布開始から早 2 年。
いよいよささみのほん 3 を Archives にマージすることとなりました。
本のタイトルも 2019(1〜2) を削り、再出発感を出して心機一転です。
思えばささみのほん 3 のあとがきには

圧倒的な締切駆動！！ 圧倒的な進捗！！

と書かれていますが、未だに圧倒的な締切駆動感が残る製作状況となっており、進歩してないなあといったところです。
あ！ あとがきが togakushi さんからじゅんじゅんに変わったので進歩というか後退 (げふんげふん
それはさておき、タイトルもあらたになりましたが、読んでくれた方の書棚 (電子の場合書棚って言うんですかね) にずっと残っていることが出来ればいいなと思います。

謝辞

3 をマージすると決めてから短期間でまとめあげてくれた togakushi さんに感謝！
表紙データを提供してくれた silpheed さんに感謝！
同人部としてダウンローダシステムのメンテに携わってくれているしののめさんに感謝！
この本を手に取り読んでくれた、あなたに感謝！！

真のあとがき

ささみのほん 4 も Archives にマージされ、ページ数がとんでもないことに。
これはお買い得すぎる。買うしかないのでは。

謝辞

この本をいつも読んでくれる、あなたに感謝！！

ささみのほん **Archives**

2019 年 7 月 27 日　初版第 1 刷　発行
2021 年 7 月 10 日　第 2 版第 1 刷　発行
2023 年 5 月 20 日　第 3 版第 1 刷　発行
制　作　ssmjp 同人部

アウトプットしないのは
知的な便秘