月刊

本誌正式名称は「シェルスクリプトマガジン vol.28」です

Shell Script Is The Supreme Glue Technology
FOR THE SOPHISTICATED SHELL SCRIPTERS

世界で唯一の

シェルスクリプトマガジン

よりぬき版

2015 8 August Vol. 28 ¥500

巻頭 TechLION再録拡大版

ゲームを取り巻くインフラ技術を語る！

伊勢幸一×清水 勲×伊藤 学×竹迫 良範

目指せシェル女子！
～具殻高校、パソコン部の日常～

Ploneシンポジウムで聞いた
CMS比較とPloneが
大学で人気な理由

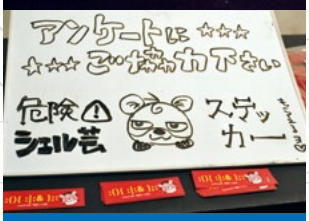
現役PG社長がプログラマーに期待する2つのこと

法林浩之のFIGHTING TALKS

ITエンジニアのためのマーケティング入門

Artwork created by Koji Ishizuka. © 2015 Zie's Bank .Co,. LTD All rights reserved.

CONTENTS



世界で唯一の
月刊シェルスクリプトマガジン
2015 August vol.28

04 **TechLION再録 拡大版**
伊勢幸一、技術との出会い。ゲームを取り巻くインフラ技術を語る!

13 **漢のUNIX　第24回**
後藤大地

20 **縁の木、育てよう　第11回**
白羽玲子

22 Ploneシンポジウムで聞いた
CMS比較とPloneが大学で人気な理由

24 **ユニケージ開発手法コードレビュー　第17回**
大内智明

29 **姐のBENTO**

30 **法林浩之のFIGHTING TALKS　第15回**
法林浩之

32 **現役PG社長がプログラマーに期待する2つのこと**
武井康浩

34 **スズラボ通信　第20回**
すずきひろのぶ

38 未来に活きる!現場で使える!
データモデリング　第18回
熊野憲辰

42 **ITエンジニアのためのマーケティング入門　第5回**
水間丈博

44 **IPv6新時代を体感しよう!第9回**
波田野裕一

48 **人間とコンピュータの可能性　第28回**
大岩元

50 中小企業診断士が解説する、超実践的な会話術!
円滑コミュニケーションが世界を救う!　第6回
濱口誠一

52 ビジネスシステムイニシアティブ協会 例会レポート
JT IT部長 引地久之氏の挑戦とは

54 **中小企業手作りIT化奮戦記　第18回**
菅雄一

57 **めざせシェル女子!　第4貝**
～貝殻高校、パソコン部の日常～
ちょまど

63 **階層**
桑原滝弥

65 **３つの「がくりょく」**
シェル魔人／編集後記

ちんじゅうちゃん

本誌名称について
vol.21より「USPMAGAZINE」から「シェルスクリプトマガジン」に、変更となりました。バックナンバーは「USP MAGAZINE」としてお求めいただけます。
本誌正式名称は「シェルスクリプトマガジンvol.28」となります。

燕去月
運用の仕事をなくすための団体、MSP協会とは？
世界で一番実行されているパソコンゲームとは？
TechLION再録を巻頭拡大版でお届けします。

TechLION
For Independent Engineers

伊勢 幸一、清水 勲、伊藤 学、竹迫 良範

伊勢幸一、技術との出会い。ゲームを取り巻くインフラ技術を語る！

聞き手：法林浩之、馮富久　まとめ：シェルスクリプトマガジン編集部　撮影：Tomocha

5月25日、六本木 SuperDeluxe にて TechLION vol.21が開催されました。
第一部 獅子王たちの夕べでは
日本 MSP 協会事務局長・テコラス株式会社の伊勢幸一氏にお話しいただきました。
ユニークなデータセンターと日本 MSP 協会がすすめる IT 業界活性化の妙案に注目です。
清水勲氏、伊藤学氏、竹迫良範氏をゲストに迎えた第二部 ジャングルバス .com では、
ゲームと IT をテーマに海外での運用からチート対策まで多彩なトークが繰り広げられました。

第一部　IT サファリパーク

久しぶりの獅子王登場、各種団体のなんでも屋

法林：第一部は久しぶりの獅子王たちの夕べです。今回お招きしたのはテコラス株式会社の技術研究所所長でいらっしゃいます、伊勢幸一さんです。皆様拍手でお迎えください！
伊勢：よろしくお願いいたします。
法林：今日は「技術と IT ジョブと私」というタイトルでお話ししていただきます。
伊勢：皆さんよろしくお願いします。では簡単に自己紹介を。伊勢幸一です。テコラス株式会社の技術研究所所長をやっております。何やってるんですか？とよく聞かれるのですが、テコラス社外の仕事が多く、大体は、IT 業界の任意団体といいますか、そういったところで雑用係をやっています。クラウド利用推進機構とかですね、オープンクラウド実証実験 TF、Open Compute Project Japan…まぁこんなことをやっています。
法林：日本 MSP 協会で事務局長もやられているんですね。ある団体では会長をやり、ある団体では事務局長をやり…結構すごいと思います。
伊勢：基本なんでも屋なので。今までに転職を3回しておりまして、新卒で入社したのは日立設備エンジニアリングという日立系の設備会社でした。機械設計をやっていたんですよ。その後、デジタルテクノロジーという Sler を経てスクウェア（現スクウェア・エニックス）というゲーム会社に行きました。今日テーマが「ゲーム」だということなんですが、実は私、ゲームはあんまり関係なくてですね、たまたま勤めた会社がゲーム会社だったというだけです。スクウェアの後は2005年にライブドアに入社しました。ライブドアは社名が変わったりしていますが、この10年間は同じ団体なんです。ということで、現在はテコラスに所属しています。

座ったままでデータのやり取りができる事が衝撃だった

伊勢：「技術と IT ジョブと私」という事ですね、インターネットと私がどう関わっているかというお話。WIDE が発足した直後、機械工学屋から IT 業界に移ったというのが1989年です。
法林：そこで伊勢さんとインターネットがつながった、という訳ですね。
伊勢：それまでインターネットとか知ら

伊勢 幸一さん

なかったですからね、正直。89年ってまだ日本にネット普及してないですから。
法林：そうですね。
伊勢：まだ FTP もできなかったので、UUCP メールとか…
法林：わかんないですよね、最近の人（笑）
伊勢：GNU の GCC とかもらうときにリクエスト出して、GCC の値を分割して、uuencode してもらって、メールもらって後で繋げる、みたいな。
法林：何十通かくらいになりますよね。
伊勢：そんなことをやっている時代にこの IT 業界に来た、と。各会社で何をやっていたかというと、デジタルテクノロジーは当時、SIer 的なことをやっていまして、富士通 S ファミリとか、松下電気のソルボーンとか覚えていますか?
法林：ソルボーン覚えてます。
伊勢：あとは DECstation とか SONY NEWS など、国内で調達して周辺機器を付けて流通させるということもやっていました。デジタルテクノロジーではネットワーク周りやテクニカルサポートなどをやっていました。この頃は、とにかくネットワーク関連が好きだった。以前は機械工学屋が機械工学をしていて、データは2分の1のオープンリールかフロッピーでやり取りしていたんですが、この業界に入って座ったままでデータをやり取りできるというのがすごい衝撃で、とにかくネットワークが好きで好きでしょうがなかった。
法林：なるほど。

スクウェアへ転職してからプログラミング好きになった

伊勢：スクウェアに転職したタイミングはファイナルファンタジー 7 の開発が始まった時でした。2D から3D に変わって、シリコングラフィックスのコンピュータが使えるエンジニアが必要だという事が求人広告に書いてあって、普通に面接を受けて採用していただきました。FF7ではひたすら開発環境の構築を行いました。グラフィックデザイナーの人達が絵を描くときにはやっぱり Macintosh を使うので、そのグラフィックを UNIX-WS や Windows 環境と共有する必要がある。Cap60サーバというものがあったのです。
法林：そういうものを用意していた、と。
伊勢：そのころから運用ツールを自分で書くようになりまして、スクウェアに入る前は C 言語と Bourne Shell くらいしか使っていなかったんですけど、この頃に色んな LL とか、JAVA とかを使って作るようになりました。この当時は、とにかくプログラミングが好きだった。AWK とか。この本が最近出たんですけど…、今日この会場に、著者が4名居るとか、5名のうち4名（笑）。…すごいですよね。
会場：笑

伊勢：1997年に、「アメリカに行け」と言われてですね、SQUARE USA という会社が立ち上がりました。この会社は、映画製作会社・ゲーム開発会社ということですが、誰かが「俺ハワイに住みたいな」「映画作りたいな」って言ったので、ハワイで映画を作ろう!となったんです。
馮：希望がそのまま。
伊勢：そうです、希望がそのまま実現してしまうというすごい会社でした。それから LA にもオフィスがあって、LA とホノルルを行ったり来たりしていました。何をやったかというと、制作スタジオの立ち上げです。誰もいなかったので一人でやったんですけど。
法林：一人で全部やったんですか!（笑）
伊勢：若者が倒れたんです。100人分の21inch の tube モニタを一人でセットアップするつらさ!
会場：笑。

ツールを作っていたら大規模分散システム好きに

伊勢：この頃に色々自作するようになりまして、当時ラックマウント型のサーバってなかったんですけど…
法林：自作したんだ!?
伊勢：自作しました。筐体から中身まで全部自分たちで…筐体は私もともと機械設計屋なので自分で設計図面を描いて業者に渡しました。2U のね。
法林：型を起こしたんだ!
伊勢：他の自作ツールとしては、DLT のジュークボックス用のバックアップシステムとか…バックアップツールのソフトウェアって出来が良くないんですよ。
会場：笑
伊勢：まぁ自分で売ってたからわかるんですけど。自分でアーキテクチャをスライドして、安全なバックアップシステムを作って運用していました。あとはレンダリングキューディスパッチシステムとか、スナップショットファイルシステムとか、必要なツールは作っていました。
法林：無いから作るという事になったんでしょうね。
伊勢：いや、あるんですよ。
法林：あ、さっき言ったようにモノが良くないということですか?
伊勢：モノは良くないし、金がかかるんですよ。製品の質もダメだし、やっぱりライセンスポリシーがすごくダメなんです。この頃のはとにかく大規模分散システムが好きだったと。この後日本に帰って play online というのをやり始めました。DC の設計と運用などをしていました。
法林：だいぶ DC 屋っぽくなりましたね。
伊勢：そうですね。そしてこの頃初めてトランジットやパブリックピアリング、プライベートピアリングを意識し始めました。最後に株式会社ライブドアですが、さくっと行きます。
法林：はい。

伊勢：後の株式会社データホテル、現在のテコラス株式会社です。以上、さて本題です。
会場：笑
法林：ラスト10分で本題に入る！

お金のかかる設備は、金を稼がない

伊勢：2013年春頃にとある北の国から、自然エネルギーを活用したDCを作らないか?という話がありました。モジュールタイプで、20ラック1モジュールで、横にどんどん長く繋げていくというDCを作りました。形はドーム型なんですが、これは、雪が積もると雪下ろしがとても大変なので、雪下ろししなくても、ずるずるずるずるっと横に落ちる仕組み。屋根のすぐ下に排気が溜まるようになっていて、その熱で屋根と雪の間に水ができて滑って落ちる、というスタイルです。
法林：へー！そういう設計なんだこれは。
伊勢：ドーム型は強度と材料のコスト費が一番少ないらしいです。面白いのは、上に空調があって、下にサーバがあるんです。普通DC屋さんはサーバのラックの上に水回りがくるというのは嫌がるんですけど、別にサーバ濡れたって構わないじゃん、という発想です（笑）
会場：爆笑
伊勢：サーバを大切にしようとすると、DCコストとか運用コストってあがるんですよ。
法林：そういうことですよね。
伊勢：サーバ壊れても良いって思えば、運用コストや建設コストって安くできるんです。そして、北の国なので、夏以外は全部外気で冷却します。夏の30度超えるときだけ雪をつかって冷やすという仕組みです。とまぁこういうDCを作ろうと試算してみました。ところが、投資回収に10年以上という計算に！何が問題かというと、1ラックあたりのDC建設コストが非常に高いという点です。ラックを立てる付帯設備にコストがかかる訳ですが、この付帯設備は金を稼がないわけですよね。ユーザがお金を出すのは、ラックの中にあるサーバなんです。中身のサーバよりも建設費の方が高い！だったらDCって、ラックとサーバだけでいいんじゃないか?と思い始めたんです。
法林：おお！
伊勢：わざわざラックとサーバを置く為にDCを建設するのではなくて、もともと電源と回線があるところに、ラックとサーバを置きにいけばいいんじゃないの、と。これが所謂ホスティング事業者の発想なんですよ。結局、箱作った方が負けなんじゃないか?となって今進めているのが、付帯設備の無いDCをいかにして作るか。
法林：なるほど。ほう…
伊勢：ということで、MovableのDCプロジェクトというものを現在進めております。
法林・馮：へー
伊勢：コード名S.A.Cといいます。
法林・馮：Stand Alone Cabinet
会場：笑。

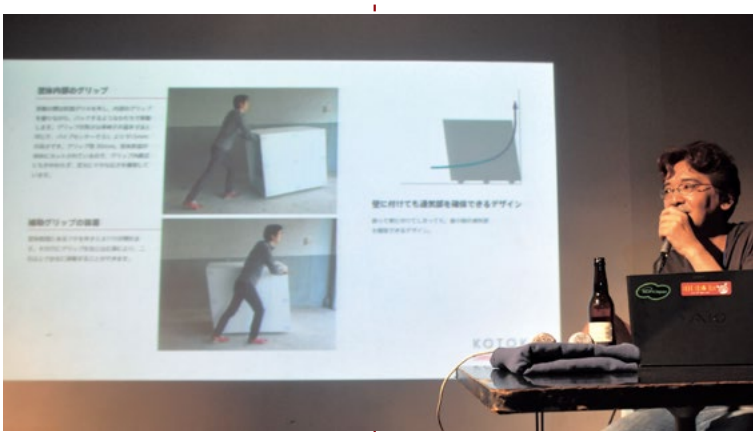
運用の仕事をいかに無くしていくか

伊勢：このスライドにでているのがハーフラックのStand Alone Cabinetです。何が違うかというと、ラックが斜めになっていて、この斜めになっている内側に取っ手がついていて、下に可動式の車輪がついていて、サーバを全部搭載しても、ちゃんと運べます。横幅が、一般マンションのエレベーターの幅より小さいので、普通のエレベーターで運べると。なので普通のマンションとか普通のオフィスの中に設置できると。
法林：すごいな
伊勢：問題は、点在するS.A.Cをいかに需要に応じてデプロイして提供するか、という技術が必要で、ベアメタルプロビジョニングや分散リソースの運用技術がキモです。それで去年の11月に日本MSP協会を設立しました。何をやっているかというと「MSPの仕事をいかに無くしていくか」ということです。
法林：そうなんですか！？
伊勢：はい、自動化とか、運用設計ガイドラインとか、どんどん標準化したりとか、ツールとか技術をどんどん公開、共有していって、運用エンジニアを無くしていこう、という協会なんです。
法林：ほー！
伊勢：なので、MSP協会というのはMSPの事業をを活性化させるというのよりは、これまでやってきたMSPのオペレーションをどんどん無くしていこうという、そういう協会なんです。そして、需要にマッチしたサービスを提供していこうと。既得権者に刈り取られるだけでは、いつまでたってもIT業界は活性化していかないので、やっぱりIT業界はIT業界で、正しいことを正しいように考えて提供していこうということで活動しています。ということで、定年まで後数年しか無いんですけれども頑張っていこうかなと思います。（笑）以上で私の話は終わりです。ありがとうございました！

第二部　ジャングルバス.com

大人気ゲーム「モンスターストライク」

法林：では第二部に入りたいと思います。ゲームとITに関する3名のゲストをお呼びしました。伊勢さんにもコメンテー



ターとして入っていただきます。
伊勢：よろしくお願いいたします。
馮：早速トップバッターの mixi の清水さんにお話しいただきたいと思います。よろしくお願いします。
清水：「モンストを支える IT インフラ」というタイトルで、ちょっと大げさ…。
法林：いやいや
馮：今までにあまり表には出ていないお話ですよね。
清水：そうですね。私は株式会社ミクシィの、モンストスタジオというところに所属しております。
法林：部署名なんだ。
清水：モンスト専門の部隊ですね。
伊勢：エニックスにもドラクエ課ってありましたよ。
清水：ゲーム業界はそういうの多いのかもしれませんね。現在入社四年目でもともとは SNS の mixi のインフラ面をずっとやっていました。モンストが出始めて、これはちょっとヤバいぞっていう時にジョインして、今に至ります。「モンスターストライク」と、最近リリースした「モンストスタジアム」のサーバ運用をやっています。モンストはもちろんやっていますが、モンハンも大好きです。モンつながりで（笑）
会場：笑
法林：モンハン好きな方も多いですよね
清水：多いですね、今日の昼休みもやりました。ほぼ毎日やっているくらいです。自己紹介はこのくらいにして、mixi の事業について知らない人が多いと思うのでご紹介します。モンストは、エンターテイメント事業ということでやっています。また、mixi は昔から SNS をやっていて、メディアプラットフォーム事業として、mixi や、Find Job!、あと最近だと nohana など色んな会社がくっついてサービスを展開しています。
法林：分け方がモンストとそれ以外だよね。
清水：完全にそうなっちゃってますね。
伊勢：mixi さんはもともと SNS が母体じゃないですか、母体とそれ以外が一緒になってるんですね。それでモンストがさらに…。
清水：そうですね、モンストがさらに別格になっちゃったという感じですね。

で、「モンスターストライク」とは、ということなんですけど、たくさんいるキャラクターで、ゲームをやっていきます。2013年10月10日に正式リリースし、先に iOS 版が出て、次に Android 版が出ました。現在は日本が一番大きな規模なんですが、台湾、香港、マカオ、中国、北米、韓国という国でもグローバル展開しています。
法林：各国語版があるってことですね？
伊勢：運営は一カ所なんですか？
清水：それぞれの言語で、運営は一カ所です。
伊勢：一カ所で運営をしていて、各国にサポートが居るような感じですか？
清水：はい。全部会社の中でやっている感じですね。世界累計利用者数は3,000万人を突破しました。
法林：すごい。
清水：それから、ゲームの特徴を簡単にご説明します。「ひっぱって遊ぶ」、というのがメインのゲームです。4人で協力プレイができるのが売りです。あとは、モンスターを強化・進化させたり、ひっぱってぶつけるので、ぶつけたときに何が起こるか、など。

インフラエンジニアとしてスマートフォンゲームを支えていくためには

清水：ここからちょっと技術的な話。どんなインフラなの？っていうところは意外と言ってこなかったので、ちょっとご紹介したいと思います。まずインフラ構成ということで、日本版はオンプレと AWS の併用です。AWS の Direct Connect というのを使って、自社で DC とつないで提供しています。ただ、AWS をがっつり使っているかというと、実はそうではなくて、意外とオンプレ比率が高い。本番はほとんどオンプレです。リリース初期は AWS がメインだったんですけど、徐々にオンプレに移していった形です。これは特徴かもしれないですね。
馮：これは何か理由があるんですか？
清水：圧倒的にオンプレの方が性能が出るんです。あとはコストパフォーマンスがすごく良いんです。実はマシンを買った方が安い。
法林：クラウドは実は高いんですね。
清水：もともと SNS の mixi がずっとオンプレで運用してきたので、その辺のノウハウがあるからという点もあります。ただ、海外版や、リリースしたばかりのモンストスタジアムは実は AWS だけだったりします。そこは分けていて、負荷がまだないものとか、これから徐々に増えたら移していく感じになるのかな。
伊勢：スタートアップとか、予測できないところは AWS でちょろっとやっておいて、バーンと当たったらオンプレの方が効率いいよということですね。海外で当たったら、海外もオンプレにするんですか？

清水 勲さん

清水：海外は現地の問題があると思うので、そこはわからないですね。AWSのままでいく可能性高いです。次にサーバの構成ですが、物理サーバはEC2でOSはUbuntuです。ディスクはSAS HDD、SSD、ioDriveで、最近はほとんどSSDにしています。アプリサーバを200台くらい出しています。日本版だけです。開発言語はRubyで、フレームワークはPadrinoを使っています。

馮：開発言語がRubyっていうのは、mixiが変わったな、という点ですよね。

清水：そうですね、mixiといえばPerlという感じでしたからね。それから開発スタイルですが、GitHubでPRベースの開発をしていて、1日に何回も本環境にデプロイします。開発環境はMac OS Xが基本です。

法林：みなさんOS Xで開発するんですね。

清水：はい、もうみんなMacですね。コミュニケーションはHipChatを使っています。これもAWSのホスティングで動かしています。それから運用ツールですが、グラフはCloudForecastを使っています。また、アラートはずっと昔から使われている、Nagiosなんかを使っています。集計はFluentdとかですね、あとプロビジョニングはChefですね。

馮：傾向はあるんですか、例えばmixiは年末年始にガッとあがるとかあるじゃないですか。

清水：ゲームとSNSは全く違うと思っていて、特に昼はゲームの方がイベントやったりとかで、かなり伸びています。あとゲームとSNSの違いは、夜のピークタイムが若干ずれているかなと。ゲームは0時をすぎると一気にやめるとか。

伊勢：サラリーマンの活動時間と一致していますよね。12時にお昼休みに入って携帯を出しゲームを始める、そして、家に帰って18時から22時くらいまでやると。そのあとは、翌日があるので寝る。

法林：そうですね、面白い。

清水：インフラエンジニアとしてスマートフォンゲームを支えていくためには、クラウドに対する考え方がいくつかあるのかなと思います。なんでもクラウドが良い！みたいな風潮が最近あるんですが、悪いところもあったりします。そしてスマートフォンの最適化で大切なのは、レスポンスを早くすること。スケールする設計・構成を出来るだけ最初から考えておくこと。Chef、Ansibleなどのインフラのコード化や、データの増え方、DBの分割の可能性、データの削除方法など、最初に考えておくと後で楽できる。まとめると、ゲームリリース当初のインフラはとっても大事ですと。急激なスケールアウト、スケールアップが求められるのがゲームならではなのかなぁと思います。スケールできないとゲームの存続すら危ういです。そしてクラウドの特性をちゃんと理解して使う。あとは開発体制や運用体制を強力に持っておく必要が有ります。というところで、私からは以上です。

ネットを駆使した地域活性

法林：では二人目のお話を伺いましょう。STUDIO Freesiaの伊藤さんで「INGRESSと地域活性とコミュニケーション」です。よろしくお願いします。

伊藤：せっかくなので、INGRESSやってる人どれくらいいらっしゃいますか？ちょっと挙手していただけますか？

馮：さすがにやっぱり多いですね

伊藤：青と言う人！…あれ

法林：少ないね

伊藤：緑という人…あぁ…

法林：結構偏ってますね

伊藤：敵ですねぇ…

会場：笑

伊藤：INGRESSに関しては色々活動しているので、今日はそのご紹介をさせていただきます。インフラ関係とは全然関係ない、Googleとも関係ないお話です。元は何屋さんかというと、学生時代は税理士を目指していたんですが、当時インターネットが流行っていてパソコン部に入ったのですが、Webサイト制作にのめり込んでその勢いでWebサイト制作ユニットSTUDIO Freesiaを結成したことが始まりです。今もWebをやっています。一時期制作会社に入って、某自動車メーカーのWebサイト設計などやったこともあります。最終的に社長が遊びすぎて潰してしまいましたけど…。

法林：おおお（笑）

伊藤：税理士の知識もあったので、お金の事もわかっていたんで察してはいたんです。この会社まずいな、と思って辞めて、フリーランスになりました。その数年後に大きな地震があって、Ustreamの関連のエバンジェリストなんかもやっていたので、地域の色んな所を渡り見て、ネットを使って何か出来ないかを考えていました。今は地域活性のお仕事を、特にネットを駆使した企画の提案をしつつ、デザイン学校の講師や、資格系の学校の講師をやっています。

法林：色んな事をやっているんですね。

伊藤：で、INGRESSですが始めたのはiOS版が出た2014年7月14日からです。で、何をやっているのかとう言うと、地域活性です。最近騒がれていますが、この辺のことを私も動いています。今地域ってどんな風になっているかというと、これは商店街の話ですが、人が来ないのでシャッター街が増えてしまっている。そして跡継ぎが居ないので高齢化が起きてしまっている、という悪循環になっ

伊藤 学さん

てしまっているんですね。やりたいんだけど、よそ者には好き勝手やらせたくない、という…。
会場：笑
法林：なかなか難しいですね。
伊藤：どうして「よそ者には好き勝手やらせたくない」という意見が出るのかというと、最近は地域活性をしようという NPO や企画屋さんが多いんです。ただ、実際の声を聞いてみると「一発で終わり」と。
馮：継続しない。
伊藤：継続できないものばっかりやって、収益を得る。例えば『助成金』をもらう事が目的になってしまっていて問題になるんです。INGRESS に関しては皆さんご存知の通り、地域活性のためのイベントをやっている地域があります。横須賀市は、デザインもホームページ全部内製しています。かっこいいメダルも作っているそうです。東村山市は、某ゲーム会社の方が「中の人」をやっているそうなんですが、会社とは全然関係なく、個人で東村山市と連絡を取って運営したそうです。その中で、私がなにをしているかというと・・・、INGRESS 飴って知ってます？
法林：INGRESS 飴知ってる人！おおすごい、みんな知ってますね。
伊藤：要は金太郎飴ですね。名古屋の飴を作っている工房があるんですが、ここの社長がお友達で、INGRESS のエージェントなんですね。彼がこの飴を作ってる最中の動画を UP したら、一晩で世界中を駆け巡ってしまったっていうことがあったんですが、知り合いなので「欲しいな〜」と言ったら、「じゃああげる」と。
法林：作ってくれた？
伊藤：いえ、この動画で作ったやつをちょっとだけいただいたんです。最初の頃は私が関東で配っていました。そうしたらたくさんの人が「コレを作っている所が見たい」という話になり、じゃあ見学会やろうかという事になりました。
法林：名古屋まで行ったんですね。
伊藤：名古屋まで行って、右手にカメラ、左手に iPhone で、Twitter で生中継しながら見学会をやらせてもらいました。
伊勢：ちなみにコレ通販はしていないんですか？
伊藤：通販今してます！
伊勢：あ、してるんですね。それはワールドワイドに向けて？
伊藤：ワールドワイドに向けてはしていないですね、日本だけです。飴の着色とか香料が海外では NG なものがあるんですけど、それがわからないんです。なので、送ってしまってトラブルになってしまったら困る、ということで、今は日本だけになっています。

一番重要なのは、地域が自立できること

伊藤：それから私の方の企画では商店街ですね。先ほど言ったように、商店街はなかなか自分たちで新しい事が出来ないので、私が今住んでる川崎の登戸の商店街で、地主さんともちょっと絡みが有るのでやらせてくださいとお願いしました。そして、一年に三回「はしご酒」というイベントが行われるんですが、一万円だけ予算を取っていただいたので、何か面白い事やろうということで、このときはミッションを用意して、まずそのミッションをクリアしてきてくださいと。参加者がミッションをクリアした上ではしご酒のチケットを買ってくれたら、抽選で INGRESS モバイルバッテリーをプレゼント、という企画をやりました。飴とガムもプレゼントしました。
法林：限定ミッションをクリアして来店すると、ワンドリンクサービスですか。
伊藤：はい、これは一応日本初と言わせていただいています（笑）
法林：初なんだ！
伊藤：リアルで報酬をもらえる、という事ですね。
法林：ああ、なるほどそういうことか！
伊藤：ゲーム上で何かではなくて、実際やってきた事でなにかもらえる、というのをやらせてもらいました。でもこれは3日間しかなかったのと、クリスマスだったので、結局一人しかこなかったんですけど。
会場：笑
伊藤：でも来てくれた人が居るというのは嬉しいです。じつはこのミッションは一時間くらいかかるもので、それをわざわざやってくれたんです。それからその次にやったイベントは年末から年始にかけて指定ミッションをクリアできたらワンドリンクというものでした。
法林：これは色んな店ですか？
伊藤：そうですね、この時は5店舗許可をもらえて「人が来なくてもいいじゃん、やろうよ」と温かい言葉をもらってやらせていただいて、これは二人の参加者（笑）。逆に、はしご酒の時はチケットが完売して INGRESS で確認がとれた人数は40人でした。
法林：かなり増えましたね
伊藤：2日間やったんですが、2日目は私も受付をやって、「青ですか？緑ですか？」「どこから来たんですか？」「青ですか？緑ですか？」って聞いて回ったら、意外にも、東京だとか…。
法林：ちょっと遠い所から来てくれていたんですね。
伊藤：こんなこともなければ来ないですよ、と。コレはコレとして成功でした。今後も色々な企画を準備しています。
法林：大体、ミッションをやって何かするともらえる、という感じですか？
伊藤：そうですね、やっぱり何か結果を見れるものにしないと、効果測定ができないっていう点があります。
伊藤：色んなセミナーやってる方のお話も聞いてみた所、地域活性で必要な事は、自分の地域で、普段付き合いがある人と、何をやったら何が起きて、お店や商店街にどんなメリットがあるのかをハッキリさせることなんです。そして一番重要なのは、やった内容が、後で自分たちでも出来るように。
法林：自立できる内容ってことですね
伊藤：はい、自立できる内容にするのが大事なのかなと。あとはエージェントさん（住人）とできれば仲良くしたいので、リアルの交流もやろうと。これらを

掲げて、少しずつやっています。こんなところです。

マインスイーパを自動で解いちゃう“マインちゃん”

法林：では続きまして三番目のゲスト、サイボウズ・ラボの竹迫良範さんです。竹迫さん TechLION 二回目の出場です。今日は「オンラインゲームのチート行為は悪なのか?」というタイトルでお話をしていただきます。それではよろしくお願いいたします。

竹迫：サイボウズ・ラボの竹迫といいます。どんな会社かというと、平均年齢33.8歳で男女比6:4で社内結婚もよくあります。公式クラブ活動が31部以上あって、DS 部なんていうものもあります。ゲームといえば、自分は『コロニーな生活 PLUS』という、INGRESS が出る前にあった位置ゲームなんですが。

憑：コロプラですね。

竹迫:そうです。PHS の位置情報を使って、1キロ歩くとちょっとお金がもらえる、という。こういった位置情報のゲームは、震災後にユーザーがどういう風に動いたかという統計情報だとか、何月何日にどうだったとか。例えば GW に東京ディズニーランドにいっぱい人が居たとか、そういった情報を位置ゲームは収集しているんですが、その時にチートをやろうとしていた人が結構居て、運営との戦いとかあったみたいなんです。

法林：昔からあった事なんだね。

竹迫：そうですね。例えばですよ、一日に東京→九州→大阪→北海道…とめちゃめちゃ移動してる人がいて、運営が「コレは怪しいだろう」と。

会場：笑

竹迫：普通 BAN する（※【BAN する】アカウント停止すること）んですが、色々調べると実は客室乗務員の人だったり。

法林：不可能じゃない訳だ!

竹迫：あとトラックの運転手の方とかも多かったです。

法林：強い職種があるんだね。

竹迫：あとですね、みなさんこのゲーム（パズドラ）はご存知ですか?コピーライトの画面出した事ありますか?

法林：あんまり出さないのかな。

竹迫：コピーライトにですね「Cybozu Labs, Inc.」って書かれていまして、これはガンホーさんが作っているソフトなんですけれども、サイボウズ・ラボはオープンソースのライブラリを公開しているんですが、それを使っていただいたので、謝辞の中に入っています。

一切開発には関わっていないんですけども、オープンソースじゃなくて売り上げの0.01% でも欲しかったなと（笑）

会場：笑

竹迫：私自身は、YAPC::Asia でマインちゃんというものを作りました。ここでクイズなんですが、世界で一番プレイされているゲームは何でしょう（1）パズドラ（2）モンスト（3）マインスイーパどれでしょう皆さん!?

会場：（ざわざわ）

竹迫：…正解は、ソリティアです。

会場：笑

竹迫：世界で一番実行されている時間の無駄です。

会場：爆笑

竹迫：私 Perl が好きなので、52,000バイトの Perl の正規表現のプログラムを作ったので、ちょっとデモします。

法 林：お、デモができるんですね。minechan.pl…

竹迫：これ実行すると、マインスイーパが起動して自動でマウスクリックされます。

法林：マインスイーパを自動で解いてくれるんですね。

憑：しかも登録までしてくれる。

竹迫：爆弾の無さそうなところをランダムにクリックしていって、ちょっとずつ潰していきます。画面の数字は確率を計算しているところです。これでマウスを自動で動かしてハイスコアの登録も自動で。

会場：笑

法林：すごい自動化!

竹迫：こんなものを YAPC で発表したことがあります。37秒で上級をクリアしたことがあります。Perl には Win32::GuiTest というテスト用のモジュールがあって、コレを使うとプログラムを実行するだけで、メモ帳が起動して foo、bar、baz などを打つ事も簡単にできます。他に、マウスもできます。自動でサインカーブを描くように、マウスを動かすこともできます。自動でクリックも。

伊勢：これオープンソースで公開されているんですか?

竹迫：はい、Perl のライブラリで公開されています。

伊勢：すごいですね。

竹迫：あとスクリーンショットを自動で撮るものなど。これらを組み合わせると、電卓のあるウィンドウだけを保存するなんてことが出来る様になります。ちょっと頭のいい人は、数字のボタンを画像解析すると、マインスーパーを自動で解けちゃうんです。ただこれデバッグが結構大変で、暴走問題というのがあって、キーボードの制御を奪っているんで、なかなか停止が出来ないという。

会場：笑

竹迫：どうしたら良いのかというと。USB メモリを使うんですね。autorun.inf に open=taskkill /F /IM perl.exe として、USB をさすと…。

法林：死んでくれるんですね。

会場：笑。

人間と機械の立場が逆転か

竹迫：これは応用で、ソフトウェアに脆弱性が無いかをしらみつぶしにテストできます。

法林：すごい。

竹迫：普通、正常系テストというのは、「実装」と「要望」があって、お客さんの要望が実装されていれば満足して、実装されていなければ機能不足ですよね。セキュリティっていうのは「負の要望」なんです。してほしくない事に対して、どうできるかという事なので、してほしくない動作が、ちゃんとできないというのは堅い作りなんですが、できれば脆弱性になっちゃう。昔、ファミコンの「裏技」、あれって実は、「脆弱性」なんです

よね。ゲームのチートはノウハウが溜まってないんです。セキュリティコンテストのSECCONで「サイバー攻防戦のリアルタイム可視化」というものがあって、サイバー模擬戦みたいなものをNICTさんが開発しました。サーバ6台24チームがそれぞれ攻撃をしているのをリアルタイムで可視化し攻略するという。

法林：すごいね、悪い事してるのにかっこ良く見える。

竹迫：そうなんですよね。これはNIRVANA改というシステムです。最近はですね、「振り込め詐欺」ってありますよね。

馮：ありますね。

竹迫：あれは機械じゃなくて人間に操作させるというものですが、オンラインゲームのチートが悪いと言われているのは、ゲームバランスが崩れるとか、ユーザの不公平感からです。株式市場のプログラム取引なんかは不公平感ばりばりで個人投資家とか不利じゃないですか、何でこれは規制されないのか。オレオレ詐欺はダメですけども、例えば大規模システムの導入とかで「発注していただければ20億円で夢のようなシステムを作らせていただきます。」というような”システム詐欺”…なぜこれが摘発されないのか。

会場：笑

竹迫：最近は株の取引を機械がやっていたりして、実は会社の経営権はコンピュータが握っているんじゃないかな。サーバ運用は24時間365日快適な空調設備で無停電。超高級ですよね。データセンターには、機械が故障すると人間が駆けつけてくれる。もう人間と機械の立場が逆転しているんじゃないかと。というところで、以上になります、どうもありがとうございました。

竹迫 良範さん

第三部

会場からも質問続出、15時の負荷とは

法林：三部の方に入りたいと思います。よろしくお願いします。この時間は、ゲームに関するあれこれを質疑応答で放談します。

馮：会場の皆さんからか、あるいはスピーカーの皆さん同士でも、多分聞きたいこともいっぱいあるんじゃないかな。

法林：じゃあまずは、参加者の皆さんからいきましょうか。

質問者：先ほどのスライドのモンストのトラフィックグラフで、15時に小さいピークがあったんですが、これは何でしょうか？

清水：15時に限った話ではないんですけれど、モンスト以外のゲームもそうだと思いますが、ゲーム内で色々とイベントがあるんです。「この時間にこういうキャラが降臨する」みたいな。モンストにはイベントを通知してくれる機能があって、通知があるとやはりアクセスが集中するんですね。それで、ちょこっと伸びて、少し落ち着くみたいなことがあったのかなと。

伊勢：多分それは逆で、我々IT業界の人間には分からないんですけど、一般的な会社には15時に休憩があります。

会場：笑

伊勢：休憩時間にちょっとトラフィックが上がるので、当然運営はトラフィックが上がるときにイベントを噛ませてくるんですよ。イベントがあるから入ってくるのではなく、人が増えた時にイベントを噛ますって言うのが運営の基本です。

馮：なるほど。

法林：僕らが世間の常識を知らなかったみたいな話だ。

清水：おやつ休憩みたいのがあるんですね。

伊勢：おやつを食べるかどうかは知らないけど。

法林：休憩って言うのは確かにあると。次、行きましょう。

オープンソースとプロプライエタリ

質問者：オープンソースが色々と便利になってくる中でも、やっぱり業務によっては、オープンにできない部分、プロプライエタリのものを使わないといけない、ここは外せないというのがもしあれば、教えてください。

竹迫：まあ、ExcelとWordですよ。

会場：笑

馮：外せない？

竹迫：絶対外せないですよね、仕事できないですね。

法林：伊勢さんは、何かないですか？

伊勢：何でしょうね。基本的に無いですけど。自分のスキルが及ばないところとか、責任取りたくない部分は、商用のパッケージを使うっていうのは基本ですね。

法林：ゲームの開発って、昔だと、商用のソフトウェアが多かったんですか？

伊勢：僕はスマートフォンとかではなく、コンシューマーのゲーム機だったんで、その場合にはSDKはプラットフォームが提供するものを使わなければいけない、という決まりがありました。

法林：オープン・テクノロジーじゃなかったんですね？

伊勢：いやSDKは提供されるライブラリですけど、開発環境はほぼオープンソースですね。

法林：ああ、そうだったんですか。じゃあそこは割と前から、オープンだったと。

伊勢：システムの運用とか業務システムで、商用のパッケージを使うというのはほぼ無いです。ただ、管理系の方たちはシステムを外注して使うという。

法林：そういう感じだったんですね。清水さんは何かありますか？

清水：そうですね、モンストでいうと、さっき上げたソフトウェアは全部オープンソースです。ずっと色々考えていたんですけども、プロプライエタリのものはないなと思っています。しいて言うなら、さっきのExcel、Wordに近いですけど、OSですよね。そのクライアントのOS。Mac OSだったりWindowsだったりです。

法林：なるほど。では次に行きましょうか。

サポートとバランス

質問者：ゲームという、ユーザーに近いジャンルの中で、サポートの面でなにか気をつけていることや注意していることがあれば教えてください。
法林：ユーザーサポート！
清水：そうですね、私はサポートの方はあまり見てないんですが、サポート用の開発メンバーがいて、「こういうツールがないと、効率よくサポートできない」という時に手作業ではとても追いつかないので、そういうツールを開発する部隊が居ます。あとは当然、窓口でユーザーさんと直接やり取りするメンバーがいて、そこは体制として必須ですね。すごく大事にしているところです。
法林：なるほど。
伊勢：サポートって、トラブルに対するものと、ゲームのバランスとかシステムとかに対するクレームの二つがあると思います。やはり、真摯というか迅速に対応するのは当然ですが。あの、世のゲーマーが「バランス、クソ。」と、ですね…。
会場：笑
伊勢：それに対して、あんまり気にし過ぎると、本来のゲームバランスが崩れてしまいます。某掲示板とか、某 SNS で、炎上してる案件とかあるじゃないですか。これは、某外資系ベンダーの人が言っていたんですけど、叫んでる人って全体の、10% 以下らしいんですよ。その10% のユーザーの意見をゲームのシステムに反映すると、逆にゲームのバランスが崩れてくるので、世の中の声の大きい人の言うことを聞くんじゃなくて、全てのユーザーがどう思っているのかを冷静に判断して、バランスの調整をかけたりする必要がある。サポートの人の心が弱って「これ対応しちゃおっかな」で調整したりとかするじゃないですか。逆にそれが他のユーザーにとってはに良くない。
法林：やっぱり、ブレない心みたいなものが求められるんですね。
伊勢：まあ、ブレないというか、誰が言ったかじゃなくて、何が正しいのかっていうのを根拠に、ゲームバランスの調整や運営をしてくのが、やっぱり必要かなと思います。

チートの定義と判断基準

質問：ゲームの中でチートになりやすい、という点での定義や判断基準はなんでしょうか。
竹迫：数年前まではアイテムのプレゼント機能というのが結構普通にあってですね、普通のユーザーがアイテム交換したりできたんです。でもそういう機能があると、リアルマネートレードなんかも出てきて、裏で「お金を振り込んだら、このアイテムをあげます」みたいな感じになってしまったので、最近はそういうアイテムのトレードとかってないはずですね。ソーシャルゲームなんてほとんどない。
法林：あんまりないね。
竹迫：なくなってますね。だから、段々ゲームの作り自体が、自動でアイテムをとっても、交換できなくなるとか、そういうのが増えてきているはずです。そういう仕組みで自動化のモチベーションをなくしていくという感じです。
法林：なるほど。
竹迫：結局そういうチートでアイテムを増やしても、もともと仮想アイテムって無いものですから、別に運営業者は減るもんでも何でもないんですね。
会場：笑
竹迫：では、何のためにそういう対策しないといけないかというと、ユーザーの不公平感ですね。たとえば、レアアイテムが複製できるという問題があったとしても、別に、複製ができたって損をする人はいないわけです。
馮：ただ、手に入らない人からすれば、なにやってるのと。
伊勢：そうですね。不公平感がすごく強まってくると、結局正当にお金を払ってゲームをしている人たちが離れていってしまうので、運営している側としては、ゲームの寿命が短くなってくるので不利益がありますよね。
竹迫：まさにおっしゃる通りです。
法林：そういった対策はしようと。
伊勢：そうですね。せっかく苦労して作ったオンラインゲームなんで、なるべく長い間稼ぎたいわけです。
法林：もちろんそうでしょう。
伊勢：チートされてもリアルマネーは減らないんだけど、自分たちが将来稼げるであろう利益がどんどんどんどん寿命が短くなってきちゃうんで、運営の方は必死になってチートを防ぐわけです。

オープンソースにしちゃうとだめなもの

清水：さっきの質問で補足というか、プロプラエテタリのソフトのことなんですけども、開発する上でのツール的なところで、たとえば HipChat とか、SaaS で提供されているものは比較的オープンソースではないというか、オープンソースにしちゃうとだめなところはあります。
法林：そういうものはありますね。
伊勢：パッケージで購入するというのはないけど確かに SaaS は使いますよね。
法林：そういうのが増えてる。
竹迫：今年は Slack とか使ってる人が増えてるみたいですね。
伊勢：Slack とか普通に使っていますね。
法林：また、そういうオープンソースでないものも、良いものは使うというのが今は多いんですかね。
伊勢：有償のアプリケーションの使い方もパッケージで購入してオンプレで使うんじゃなくて、サービスとして提供されてるものを、そのまま現金などで払って使うという利用形態は確かに残ってる。そっちの方に逆にシフトしていますね。
法林：そういう手はありますよね、ありがとうございます。ではこんなところで、あっという間にお時間です。このあたりで三部も締めたいと思います。出演者の皆さん、どうもありがとうございました。

BSDコンサルティング株式会社　取締役　最高技術責任者　後藤大地

self を実装してみよう！

データ処理に欠かせない self コマンド 1

ユニケージ開発手法で頻用されるコマンドのひとつに self がある。これは awk '{print $3,$10,$1}' などの機能に相当するもので、指定した列のみを抜き出すという動きをする。

たとえば、次のようにアカウントデータを空白区切りのデータに変換して表示させてみよう。

プロンプト 1.1 空白区切りのデータを用意

```
% tail +3 /etc/passwd | head -10 | tr ' ' '_' | tr ':'
' '
root * 0 0 Charlie_& /root /bin/sh
toor * 0 0 Bourne-again_Superuser /root
daemon * 1 1 Owner_of_many_system_processes /root /
usr/sbin/nologin
operator * 2 5 System_& / /usr/sbin/nologin
bin * 3 7 Binaries_Commands_and_Source / /usr/sbin/
nologin
tty * 4 65533 Tty_Sandbox / /usr/sbin/nologin
kmem * 5 65533 KMem_Sandbox / /usr/sbin/nologin
games * 7 13 Games_pseudo-user /usr/games /usr/sbin/
nologin
news * 8 8 News_Subsystem / /usr/sbin/nologin
man * 9 9 Mister_Man_Pages /usr/share/man /usr/sbin/
nologin
%
```

出力結果をさらに self １３４６のように絞り混むと、1列目、3列目、4列目、6列目のみが出力されるようになる。データの選択、データの絞込みはユニケージ開発手法の実装面での根幹をなしている。それが故に self コマンドはとても重要な役割を担っている。

プロンプト 1.2

```
% tail +3 /etc/passwd | head -10 | tr ' ' '_' | tr ':'
' ' | self 1 3 4 6
root 0 0 /root
toor 0 0 /root
daemon 1 1 /root
operator 2 5 /
bin 3 7 /
tty 4 65533 /
kmem 5 65533 /
games 7 13 /usr/games
news 8 8 /
man 9 9 /usr/share/man
%
```

self コマンドの動作は UEC（https://uec.usp-lab.com/）にマニュアルがあるのでそちらを読んでもらえればと思う。self コマンドはユニケージ開発手法の中でも特に頻用されるコマンドのひとつで、このコマンドの動作速度はシステム全体の速度に大きく関与してくる。

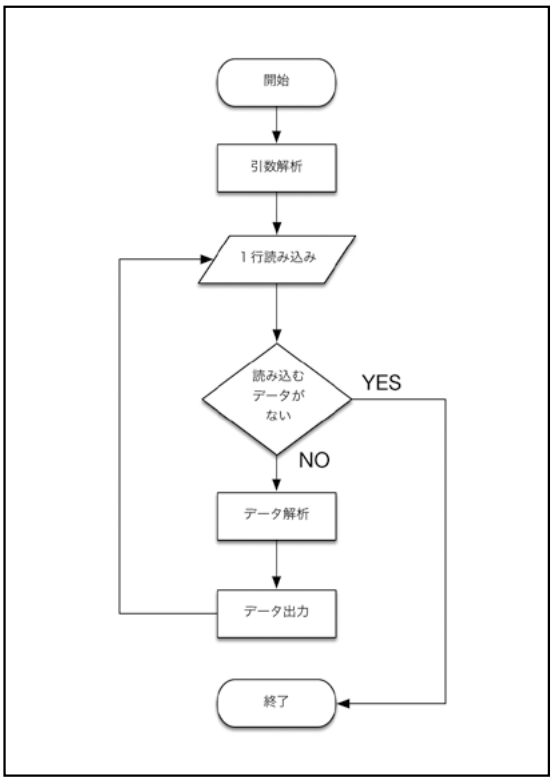
代表的なコマンドだし、実用的なコマンドを実装するサンプルとしても参考になるので、ここではシンプル版の self を実装してみよう。以降で、コマンドを開発する際の考え方の順序、うまい切り捨て方、フローとデータ構造、実装の規模などを一通り示しておこうと思う。キーワードは小さく、繰り返しだ。

フローを考える 2

こうしたツールを開発するときのポイントは、最初から全部作ろうとせずに、小さい機能を実装して徐々に大きくしていくことだ。レゴブロックのブロックを作るように、機能を作り、それらを組み合わせて最後に動作するものを作ればいい。最初から大きく作ろうとするとなんだかよくわからないものになる。

まず、self の動作を考える。最初に引数を処理するはずだ。どの列を取り出すのか、どのファイルを処理するのか、それを決める必要がある。次に、指定されたファイルから1行づつデータを読み込んで、指定された列だけを出力すればよい。処理の流れを整理すると次のようになるだろう。self に限らず、テキストデータをフロー処理するコマンドは概ね基本的にこんな流れになるだろう。

図2.1 self コマンドの基本的な処理の流れ



1行ずつ読み込んで該当列だけ出力するところは、もうちょっと分けて考えた方がよい。データを読み込む、データを解析する、解析結果のデータから該当する列だけを取り出す、といった流れになる。

こまかいことは考えない 3

開発の初期の段階では、細かいことは考えない方がうまくいく。たとえば最初から次のようなことを考えて実装にとりかかったとする。

- 1行あたりの最大データ長はメモリがある限り任意の長さを取れるものとする
- 1行あたりの最大のデータ数は目盛りがある限り任意の数を取れるものとする
- 引数にファイルが指定されていない場合には標準入力からデータを得るものとする
- N/M の指定で N 列目から M 列目までを意味するものとする
- NF の指定で一番最後の列を意味するものとする
- NF-4といった表記で一番最後の列から4列目を意味するものとする
- F.N.M といった表記で、N列目の文字列のN文字目からM文字目までを切り出して出力するものとする

かなり大仰になってくるか、扱いやすいライブラリを導入することになるだろう。こういうときはまず条件を限定してしまう。例えば次のように条件の方を定めてしまう。

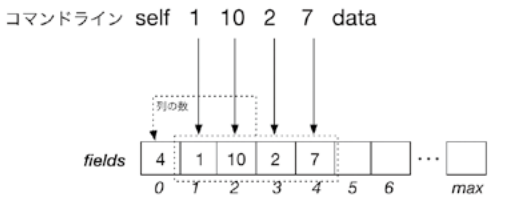
- 1行あたりの最大データ長はシステムのデフォルトに準拠する
- 1行あたりの最大のデータ数はシステムのデフォルトに準拠する
- 引数は f1 f2 f3... file とし、ファイルの指定は省略できないものとする
- フィールドは番号のみが指定できるものとする
- 引数の指定、処理されるファイルのフォーマットはすべて正しいものとし、エラーチェックはおこなわない

こういった前提を定める、条件を絞り混むことで、ソースコードを短く保ち、本質的な部分だけが書けるようになる。追加の機能や機能拡張については後々やっていけばよい。このように段階を踏んで開発の工程を切り分けることが、素早く開発するポイントだ。

データ構造を考える 4

次にこのフローを見ながら、どういったデータ構造でデータを保持すればいいかを考える。シンプルでわかりやすいものがよい。たとえばここでは、次のようなデータ構造（配列）で引数に指定された「どの列をどの場所に出力するか」を保持するとする。

図4.1



ユニケージでは列の指定は1からはじまるが、C言語の配列のインデックスは0からはじまるので、C言語側でデータを入れる場所をインデックスの1からにしている。そのかわり、インデックスの0には指定された列の数を格納するものとする。

こうしたちょっとしたセマンティックの差がコーディング時のバグのもとになることが多いので、揃えてしまうのは悪くない発想だ。

では引数を処理して、それを配列に展開するところだけをコーディングしてみよう。次のようになるだろう。strtol() 関数を使って文字列を int に変換してデータを格納している。

リスト4.2 引数を処理する arg.c

```
#include <unistd.h>
#include <limits.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sysexits.h>

int
main(int argc, char *argv[])
{
    const int arg_max = sysconf(_SC_ARG_MAX);

    /*
     * fields[0]    指定されたフィールドの数
     * fields[1]    1つ目に指定されたフィールド番号
     * fields[2]    2つ目に指定されたフィールド番号
     *    …
     */
    int fields[arg_max], i;
    char *path;

    /* 引数解析 */
    path = argv[argc - 1];
    fields[0] = argc - 1;
    for (i = 1; i < fields[0]; i++)
        fields[i] = strtol(argv[i], NULL, 0);

    /* 解析結果出力 */
    printf("file: %s\n", path);
    printf("target: ");
    for (i = 1; i < fields[0]; i++)
        printf("%d ", fields[i]);
    printf("\n");

    return (EX_OK);
}
```

プロンプト4.3 arg の動作例

```
% cc -o arg arg.c
% ./arg 10 1 3 2 /COPYRIGHT
file: /COPYRIGHT
target: 10 1 3 2
%
```

sysconf(_SC_ARG_MAX) は IEEE Std 1003.1-1988 (“POSIX.1”) で定められたインターフェースで、システムで定められている値の上限値や現在値などを取得するものだ。_SC_ARG_MAX は execve(2) ファミリーで指定できる引数の最大のバイト数を定めているので、この値を流用している。sysconf(3) はいろいろ値を引っ張ってくるのに便利なので覚えておくとよいだろう。

return() では sysexits.h ヘッダで定義されている値で終了している。任意の数字にするのではなく、システムが提供している値で統一しておけば、ほかのプラットフォームに移植するときなんかもマクロを定義したり一括置換だけで処理が完了したりするのでなかなか便利だ。

入出力を考える 5

次に入出力を実装してみよう。1行読んで、1行出力して、読み込む行がなくなったら処理を終了するコマンドのソースコードは次のようになる。

リスト5.1 1行ごとに読み込んでは出力する cat.c

```
#include <unistd.h>
#include <stdio.h>
#include <sysexits.h>

int
main(int argc, char *argv[])
{
    const int line_max = sysconf(_SC_LINE_MAX);

    FILE *fp;
    char buf[line_max];

    fp = fopen(argv[1], "r");
    while (NULL != fgets(buf, line_max, fp))
        printf("%s", buf);
    fclose(fp);

    return (EX_OK);
}
```

ここでのポイントは1行あたりの最大バイト数をsysconf(_SC_LINE_MAX) で取得していることだ。FreeBSD 10.1-RELEASE のデフォルト値は2KB になっている。大きめのデータを処理するにはちょっとこの値は少ないかも知れないので、必要があれば拡張してもらえればと思う。

実行すると次のようになる。

プロンプト5.2 cat の動作例

```
% cc -o cat cat.c
% ./cat /COPYRIGHT | head
# $FreeBSD: releng/10.1/COPYRIGHT 260126 2013-12-31
  12:24:38Z gjb $
#       @(#)COPYRIGHT   8.2 (Berkeley) 3/21/94

The compilation of software known as FreeBSD is
distributed under the following terms:

Copyright (c) 1992-2014 The FreeBSD Project. All rights
reserved.

Redistribution and use in source and binary forms,
with or without modification, are permitted provided
that the following conditions
%
```

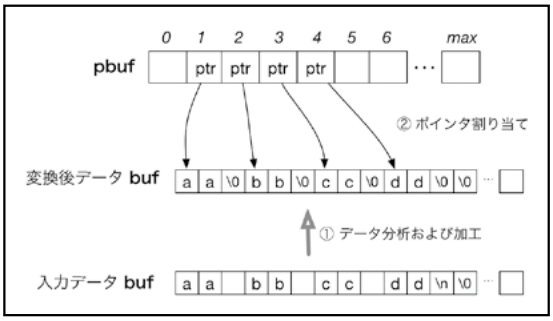
1行ごとに入出力を繰り返しているだけなので、動きとしては cat(1) と同じだ。

データ構造を考える パート2 6

次に分析と出力部分も考える。できればアルゴリズムが（実装が）簡単になるようなデータ構造が望ましい。実装を複雑にしたくない。先ほど fields[] という配列を作って、配列のインデックスとコマンドラインで指定された列の順番を対応させて格納したのだから、これがそのまま使える入力データ用のデータ構造が欲しい。

たとえば3つ目に指定された列数は2、つまり field[3] の値が2。これをそのまま使って pbuf[field[3]] のようにしたら2列目のデータ（文字列）にアクセスできる、ようにすると、実装がとてもスッキリする。言葉でまとめるとちょっと意味がわかりにくいが、図にまとめると次のようになる。

図6.1 入力データ解析およびアクセス用のデータ構造



こうすれば、たとえば、2番目に指定された列番号は fields[2]、2番目の列の文字列は pbuf[fields[2]] としてアクセスできるようになる。入力されたデータの空白をそのまま \0に置き換えることで、新しくメモリを確保することなくそのデータを使い回しているところがポイントだ。あとはそれぞれの文字列の先頭を指すようにポインタを当てはめた配列を用意すればよいというわけだ。

これを実装してみる。次のサンプルでは、pbuf[] まで作成し、あとは確認のために奇数の行だけ出力させるといったことを実施している。

リスト6.2 入力行を解析するところまで実装した cat2.c

```
#include <unistd.h>
#include <stdio.h>
#include <sysexits.h>

int
main(int argc, char *argv[])
{
    const int line_max = sysconf(_SC_LINE_MAX);
    const int arg_max = sysconf(_SC_ARG_MAX);

    FILE *fp;
    char buf[line_max];
    char *pbuf[arg_max], *p;
    int i, j;

    fp = fopen(argv[1], "r");
    while (NULL != fgets(buf, line_max, fp)) {
        /*
         * 行を分解
         *      pbuf[i] が i番目のフィールドの文字列の
         *      先頭を指すように設定
         */
        i = j = 1;
        pbuf[i] = p = buf;
        while (*p != '\0') {
            switch (*p) {
            case ' ':
                *p = '\0';
                pbuf[++i] = p+1;
                ++j;
                break;
            case '\n':
                *p = '\0';
                ++j;
                break;
            }
            ++p;
        }

        /* 奇数行を出力 */
        for (i = 1; i <= j; i++, i++)
            printf("%s ", pbuf[i]);
        printf("\n");
    }
    fclose(fp);

    return (EX_OK);
}
```

実行すると次のようになる。

プロンプト6.3 cat2の実行例

```
% ./cat2 /COPYRIGHT | head
# releng/10.1/COPYRIGHT 2013-12-31 gjb (null)
#       @(#)COPYRIGHT   8.2 3/21/94

The of known FreeBSD distributed the following

Copyright 1992-2014 FreeBSD All reserved.

Redistribution use source binary with without
modification, permitted that following wing
%
```

これで引数の分析、1行入力しての分析と加工、そこからの出力のすべてのブロックが実装できたことになる。それでは、これらブロックをつなぎ合わせてみよう。

華麗にがっちゃんこ 7

必要となる基礎技術はすべて実装したので、あとはこれらを結合して目的とする動作を実現すればよい。だいたい次のような実行になるだろう。今まで掲載したソースコードを綺麗につぎはぎしたものだ。

リスト7.1 self コマンド第一弾

```
#include <unistd.h>
#include <limits.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sysexits.h>

int
main(int argc, char *argv[])
{
    const int line_max = sysconf(_SC_LINE_MAX);
    const int arg_max = sysconf(_SC_ARG_MAX);

    /*
     * fields[0]    指定されたフィールドの数
     * fields[1]    1つ目に指定されたフィールド番号
     * fields[2]    2つ目に指定されたフィールド番号
     *      ...
     */
    int fields[arg_max], i, j;
    char *path, buf[line_max], *pbuf[arg_max], *p;
    FILE *fp;

    /* 引数解析 */
    path = argv[argc - 1];
    fields[0] = argc - 2;
    for (i = 1; i <= fields[0]; i++) {
        j = strtol(argv[i], NULL, 0);
        fields[i] = j;
    }

    fp = fopen(path, "r");
    while (NULL != fgets(buf, line_max, fp)) {
        /*
         * 行を分解
         *   pbuf[i] が i番目のフィールドの文字列の
         *   先頭を指すように設定
```

```
         */
        i = j = 1;
        pbuf[i] = p = buf;
        while (*p != '\0') {
            switch (*p) {
            case ' ':
                 *p = '\0';
                  pbuf[++i] = p+1;
                  ++j;
                  break;
             case '\n':
                  *p = '\0';
                   ++j;
                   break;
            }
            ++p;
        }

        /* 指定フィールドを出力 */
        for (i = 1; i < fields[0]; i++)
             printf("%s ", pbuf[fields[i]]);
        printf("%s\n", pbuf[fields[fields[0]]]);
        }
        fclose(fp);

        return (EX_OK);
}
```

処理速度の違いを調べたいので、1.8GB ほどあるテストデータを用意しておく。わかりやすいように1列目には1、10列目には10という値が入っているデータだ。

プロンプト7.2 速度チェック用の試験データ

```
% head data
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
% filesize data
1911685120
% gyo data
53102365
%
```

これを作成した簡易 self、awk（One True Awk; 原作者による AWK 本で解説されている awk 実装。FreeBSD 10.1 の awk はこのオリジナル awk になっている）、高速に動作する mawk で動作を比較してみる。

これは余談だが、awk の実装系としては mawk が高速だ。awk の実装系を使うときは mawk を使うというのをデフォルトにするのは、なかなか悪くない選択肢だ。

プロンプト7.3 作成した簡易 self

```
% /usr/bin/time -lph ./self 1 10 2 7 data > /dev/null
real 27.92
user 27.70
sys 0.21
      2028  maximum resident set size
         4  average shared memory size
         4  average unshared data size
      3199  average unshared stack size
       105  page reclaims
         0  page faults
         0  swaps
         0  block input operations
         0  block output operations
         0  messages sent
         0  messages received
         0  signals received
       149  voluntary context switches
       655  involuntary context switches
%
```

プロンプト7.4 awk（One True Awk）

```
% /usr/bin/time -lph awk '{print $1,$10,$2,$7}' data >
/dev/null
real 75.23
user 75.02
sys 0.20
      2600  maximum resident set size
       136  average shared memory size
        12  average unshared data size
       128  average unshared stack size
       184  page reclaims
         0  page faults
         0  swaps
         0  block input operations
         0  block output operations
         0  messages sent
         0  messages received
         0  signals received
       398  voluntary context switches
      1609  involuntary context switches
%
```

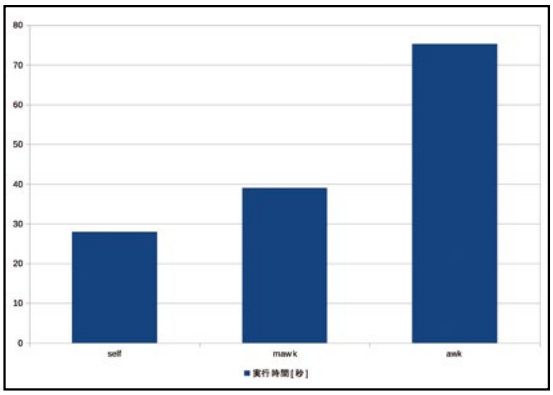
プロンプト7.5 mawk

```
% /usr/bin/time -lph mawk '{print $1,$10,$2,$7}' data >
/dev/null
real 39.01
user 37.99
sys 0.99
      2480  maximum resident set size
       136  average shared memory size
        68  average unshared data size
       128  average unshared stack size
```

```
        159  page reclaims
          1  page faults
          0  swaps
          3  block input operations
          0  block output operations
          0  messages sent
          0  messages received
          0  signals received
        210  voluntary context switches
        826  involuntary context switches
%
```

グラフで比較すると次のようになる。短い方が性能がよい。

図7.6 実行速度比較



たったこれだけ、特に高速化技巧を入れたわけでもなく、C 言語の教科書に書かれているような単純なコードだけで作ったコマンドがもっとも高速に動作している。awk と比較して3倍弱ほど早い。mawk はさすがに高速だが、それでも自作版の self の方が高速だ。

コマンドをシンプル、単機能で作成することの利点はここにもある。シンプルに開発しても既存のコマンドよりも高速に動作するものが作れる。まずこのように機能としては確実に動作するが実装はシンプル、といったものを作って、そこから必要に応じて改善すればよい。たとえばもっと速度が欲しいなら高速になるように実装すればよいし、機能が欲しいなら順次追加していけばよい。

でもこれまで繰り返し述べてきたように、もっとも大切なのは手を動かして「まずは作ってみる」ことだ。エンジニアリングとは現実との不断の対話と科学技術の絶えることのない学習に過ぎない。最新の科学を学び、エンジニアがそれを技術で現実化する。科学を応用して感じることができる何かを作る。そのとき実現できることを実現させるのがエンジニアリングだ。ハードだろうがソフトだろうが、まずは作って試してナンボということである。

さらに先へ 8

いままで説明してきた KISS の原理を超えるのは、現実がそれを超える要求をしてきたときになる。たとえばこの実装だと実行速度が遅いとか、メモリを使いすぎるとか、そういった状況になったときに次の一手として「高速化」や「省メモリ化」を模索すればよい。

そういったアルゴリズムやデータ構造は古くから研究されているし、優れた実装も多くでている。素の C でこのようになにも考えないようなコードを書いた場合、実は本質的にとても高速なのでこれを超える速度となると・・・プロセッサの専用命令を使うようにアセンブラを使う必要が出てくるのでなかなか難しいところがあるが（複雑にし過ぎるとソースコードそのものが短命になる）、高速化や省メモリ化やり方はいくらでもある。

技術に正解はないし、それは日々書き換わっていく。いつでも書ける、書き換えられる、対応できる、そしてそれをこなす知識と経験、ないしは実験を経ておく、というのが漢の UNIX 的にはとても重要だ。毎度のことながら、みなさんには是非、手を動かして実装をしてもらえればと思う。

書籍紹介

『フルスクラッチから１日でCMSを作る シェルスクリプト高速開発手法入門』

業務システムについて詳しい話をすることはできないが、本書では実際に発生したインシデントからいかにシェルを活用してシステムを構築したのかが、時系列にそって説明されている。実際のシステム開発の参考になるはずだ。また、コマンドやオプションの使い方も丁寧に説明があるので、コマンドスキルをあげたいUNIXユーザにもぜひお薦めしたい内容になっている。

引用元：http://ascii.asciimw.jp/books/books/detail/978-4-04-866068-6.shtml

これであなたもユニケージエンジニア！

ユニケージ開発手法教育講座

「ユニケージ開発手法教育講座」は、ユニケージ開発手法におけるデータ管理の方法や、オリジナルコマンドの使用方法などをハンズオン形式で具体的に学べる講座です。UNIX の基礎からユニケージ開発手法による開発プロジェクトの進め方まで、ユニケージエンジニアとしてのトータルスキルを習得できます。

http://www.usp-lab.com/LECTURE/CGI/LECTURE.CGI

- K-BASIC ユニケージ基礎編
- K-WEB WEBアプリケーション編
- K-BATCH バッチ処理編（ウェブアプリケーション処理）
- K-ARCH ユニケージアーキテクチャ編
- K-SETUP ユニケージ開発環境セットアップ編
- K-UNYO システム運用・管理編
- K-PROJECT プロジェクトマネジメント・人材育成編
- K-SQL 速習：SQL からの移行編
- K-STAT ユニケージにおける統計コマンド編

続々と新講座も充実中！

UNIX 初心者のための講座などもご用意しています。

TechLION
For Independent Engineers

http://techlion.jp/

SNS
https://www.facebook.com/usp.techlion
@techlion_staff | hashtag | #TechLION

日々新技術が誕生するITの世界。
そしてその新しい技術を追い求め、
自らを高めんと研鑽するITエンジニアたち。
「TechLION」は、
そんな技術の草原で百獣の王を目指すエンジニアたちが集う
新感覚トークライブ！
彼らの激白と酒の力で、
IT技術の面白みと深さをともに分かち合いませんか。

上記のサービス内容は 2015 年 7 月現在のものです。最新の情報はホームページにてご確認ください。

ISBN 978-4-904807-22-4
C3455 ￥500

9784904807224

USP 研究所
定価（本体 500円+税）

1923455005002