

WORD

2011.12

From College of Information Science



vol.20

目次

吉村くんでもできる自炊	03
WORD 編集部員、 パンデミックから4度世界を救う	09
エターナル餃子チケットのすすめ	15
電子の歌姫はアイドルの夢を見るか -extend-	21
GRな日々。 IX - 図書館探訪	33
カーネルコンバイル虎の巻	35
実用 Perl6	40
マニアック正規表現 I	49
正規表現で微分	60
情報科学類誌 WORD 読者アンケート	68



吉村くんでもできる自炊

文 編集部 残陽

はじめに

まさに冬、寒くて死にそうですね。皆様いかがお過ごしでしょうか。

はじめまして、coins11 の最後の良心、残陽(@zanyou)と申します。はじめての記事です。緊張しますね、ちょっとおなか痛くなってきました。

「吉村くんでもできる自炊」とは、以下のような人におすすめの記事です。

- ・一人暮らしを始めた当初は「自炊するぞ！」とやる気に満ちあふれていた吉村くん。
- ・だけど正直だるいし、なんかうまくできないから外食でいいやと諦めてしまった吉村くん。
- ・実は最初から自炊をする気などなく、誰かに作ってもらったり外食するつもりでいた吉村くん。

この記事は私、残陽の独断と偏見に基づいた「誰でもできるんじゃね？」レベルの自炊を紹介していくものです。

吉村くんとは

吉村くんは coins11 の AC 入試組であり WORD 編集部の一員。Perl と正規表現をこよなく愛する人です。この記事のタイトルを「吉村くんでも……」としたのは、私の友人の中で最も「自炊をする気がない」と自他共に認める存在だからです。

そんな吉村くんの自炊に対する姿勢を表した逸話があります。

筑波大学に入る前、吉村くんの母親は息子が一人暮らしを始めたとき、ちゃんと生活できるように「自力で食事を作るよう」に言った。

吉村くんはそれに従い自炊することにした。

彼はAmazon.co.jpで大量のうどん(乾麺)を購入し、うどんを基軸に自炊期間を乗り切ることを決める。

初日、うどんを茹でているときに鍋の中で対流が起こっていることに気がついた。

「素晴らしい！説明にはかき混ぜろと書いてあるが、かき混せるのは対流にまかせればいいな！」
まず最初の1週間は素うどん(うどんを茹でてザルにあげ、めんつゆをかけるだけ)のみで乗り切った。

しかし流石に飽き、残りの週には惣菜屋で買ってきた天かすを大量投下したうどんとざるうどんとを交互に食べた。

そしてうどんのみで3週間を乗り切ったのである。

おわかりいただけただろうか

おかわりいただけるだろうか

この他にも吉村くんはそばのみの生活やパスタのみの生活などを経験しています。

筑波大学に入った後も 1 学期中は私が作ったご飯を食べていましたし、2 学期中はほとんど外食をしていたらしいです。

吉村くんでもできる

ご飯が炊ける

まず基本ということでご飯の炊き方から。

「えっ、炊飯器を使えばボタンひとつで炊けるじゃないですか」って人がいると思います、というか私がそうです。ですがあえてご飯の炊き方を紹介するのには2つの理由があります。

- ・吉村くんは炊飯器の使い方がわからない。
- ・というか吉村くん炊飯器持ってない。

今回の記事は吉村くん基準で書いてるので、申し訳ありませんが我慢してください。

2-1, 米を研ぐ

意外と正しいお米の研ぎ方を知らない人も多いのではないでしょうか。

- 1 : ボウル状のもの(ザル+ボウルが理想)に米を入れる。
- 2 : そこに水を入れ軽くかき混ぜ、すぐに水を捨てる。
- 3 : 米を研ぐ(一定のリズムでかき回すように)。
- 4 : 水を入れ、軽くかき混ぜ、水を捨てる作業(すすぎ)を2回。
- 5 : もう一度研ぐ。
- 6 : もう一度すすぐ。

およそこんな感じです。

2-2, 米を炊く

炊飯器だと研いだ米を入れて、分量通りの水を入れてスイッチをいれるだけでOKなので、今回は釜を使った炊き方を紹介します。

- 1 : 「おぎのや」の「峠の釜めし¹」を手に入れる。
- 2 : 釜を洗い、よく乾燥させる(割れてしまう危険性があるため、すぐ使いたいときも外側だけでもよく拭く)。
- 3 : 研いだ米と水180ccを釜に入れ蓋をし、5~30分吸水させる。
- 4 : 弱火にかける。
- 5 : 吹きこぼれてくるので蓋を1cmほど開け、火は最弱に。
- 6 : 吹きこぼれがなくなったら蓋を少し閉める。
- 7 : 米の表面に水たまりがなくなったら火を止め、蓋を完全に締める。
- 8 : 15分以上蒸らす(蓋は絶対に開けない)。
- 9 : 完成。

ほら、簡単でしょう？

*1 峠の釜めし : <http://www.oginoya.co.jp/oginoya02/tougenokamameshi/index.html>

ハンバーグ

お惣菜よりも少し自炊レベルの高い「マルシンハンバーグ」を紹介します。スーパーなどで 1 コあたり 100 円ほどで売っています。調理方法は「フライパンで焼くだけ」油を引く必要もありません。表面に少し焦げ目がつくまで焼けば完成です。ケチャップやソースなどをかけて食べましょう。意外とおいしいですよ。

サラダという名のsomething

定食を頼むとよく千切りキャベツが出てきますよね。ドレッシングをかけておいしくいただけるわけですが、包丁でのレベルの千切りをするのには相当の鍛錬と修行が必要です。

そこで吉村くんでもあのレベルの千切りが作れる素敵アイテムを紹介します。
"スライサー^{*2}"と"野菜水切り^{*3}"です。



スライサー



野菜水切り

キャベツ半玉（60 円程度）で 3 ~ 4 人分のキャベツの千切りが作れます。

スライサーを使ってキャベツの千切りを作り、水を張った野菜水切りに投入してかき回す。その後水を捨てて水を切れば、おいしいサラダの完成です。

吉村くん「この装置は素晴らしい！僕や残陽の技量でも、この精度で千切りキャベツが作れるとは……うつ」

どうしましたか。

吉村くん「ちょっと指を切った……痛い……この装置はオワコンですよ」

*2 スライサー：キッチンスライサー。1500 円ほどでスライス、せん切り、つま切り、おろし器のセットが販売されている。

*3 野菜水切り：野菜水切り機、もしくはサラダスピナー。ハンドルを回すだけで野菜の水切りができるお手軽アイテム。Amazon.co.jp で 1400 円ほど。

吉村くんでもできる

まとめ

吉村くん「スライサーがあればいくらでもサラダが作れますね！キャベツはおいしいので毎日作りますよ」

それは良かったです。

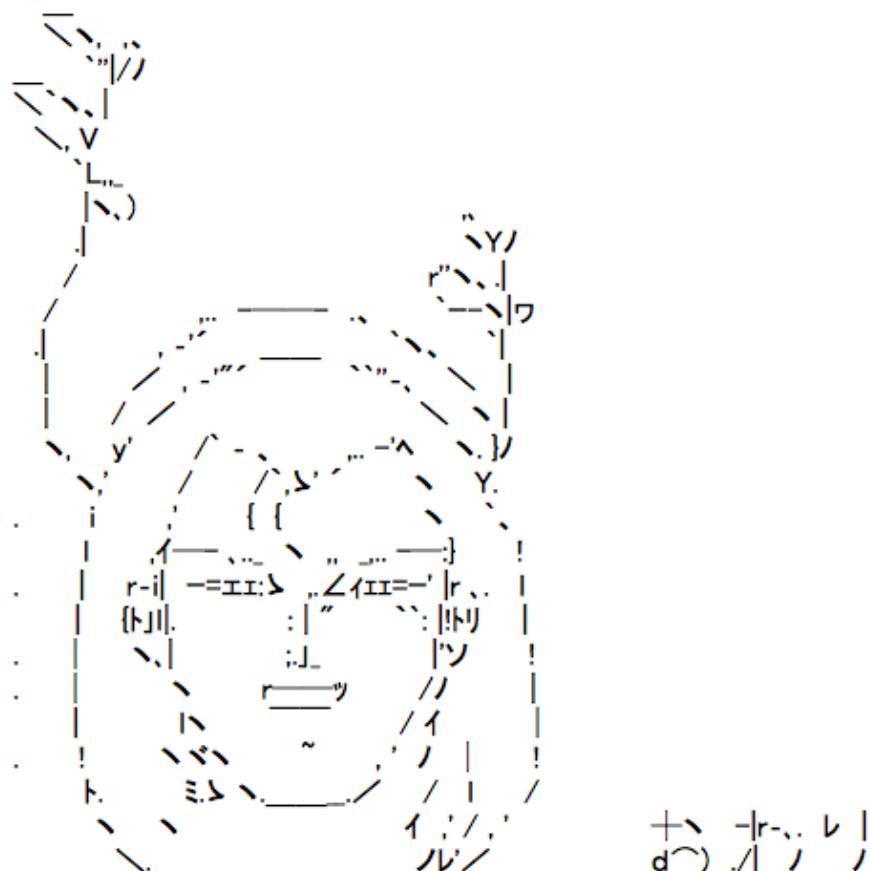
吉村くん「サラダはドレッシングを食べるためにある！（トバア）」

たくさんかけますね。

吉村くん「かけすぎた……このサラダはオワコンですね」

「」

吉村くん「人間生きているだけで満身創痍ですよ」



WORD 編集部員、パンデミックから 4 度世界を救う

文 編集部 enerick

はじめに

こんにちは。まず最初に断っておくと今回はパンデミック（テーブルゲーム）の話になります。

皆さん、テーブルゲームはお好きでしょうか。大学生という生き物はテーブルゲーム好きが多いようなので大体みんな好きだと思います。僕の友人にも麻雀というテーブルゲームを夜を徹して嗜んでいる人がいます。ところがお恥ずかしながら、僕はテーブルゲームをあまりやったことがないです（ぼっちだからやる相手がいなかったとかそういうのじゃないですよ）。

前置きはこの辺にして、今回はパンデミックというテーブルゲームの紹介をします。とりあえず、前半ではパンデミックというゲームについて Wikipedia 先輩を大いに参考にしながら軽く説明をして、後半では先日 WORD 編集部として参加してきた*1「第三回パンデミック大会」の参加記を書きます。パンデミックのことを知らない人はこのまま前半から読むと良いでしょう。パンデミックのこと知ってるしクソみたいな紹介文なんて読めるか！っていう人は後半まで飛ばすと良いでしょう。

パンデミックとは

パンデミックは一言で説明すると、新型ウイルスの世界的流行（パンデミック）を必死で阻止するテーブルゲームです。このゲームの特徴は、プレイヤー全員 VS ゲームシステムという協力プレイ方式である点です。協力プレイ方式であるということは、それすなわち一蓮托生なのです。人生ゲームやモノポリーのように他人を蹴落とし己の繁栄のみを求める殺伐としたゲームとは一味違うのです。まあウイルスという強大な敵を前にして人類が団結するのは必然とも言えましょう。つまるところ、このゲームは勝ってみんな嬉しい、負けてみんな悔しい、実にハートウォーミングなゲームなのです！

ゲームの概要に少し触れたところで、簡単なルール説明を加えます。

まずこのゲームは世界地図を模した「マップ」と、「プレイヤーカード」「感染カード」「病原体コマ」を用いて行われます。マップには 48 の都市が 4 色に分けられて表示されており、感染カードを引いた時にそのカードに記されている都市で感染が進行します。各都市の感染度は病原体コマの数で示します。同じ色の病原体コマは 1 つの都市に 3 つまでしか積むことはできず、4 つ目を積むことになった時に「アウトブレイク」が起こります。アウトブレイクが起こると、周囲の都市に 1 つずつ感染コマを積む事になります。隣の都市が 3 つの感染コマが積まれている状態だった場合、その都市でも連鎖的にアウトブレイクが発生します。1 度の連鎖で起こるアウトブレイクは各都市 1 回です（無限ループは起こらない）。1 つのターンにつき、プレイヤーは 4 回の行動をすることができます。行動には「移動」「感染コマの除去」「基地² の設置」「治療薬の作成」「手札カードの受け渡し」があります。治療薬の作成には同じ色のカードが 5 枚必要になり、なおかつ自分が基地の上に居る必要があります。

*1 僕は欠員のピンチヒッターとして、前日夜にルールのおさらいをしてから参加した。

*2 基地間を 1 手で移動できたり、カードが揃っていれば治療薬が作成できたりと、このゲームの攻略に不可欠なもの。

ぱんでみっく!!

プレイヤーは自分のターンの最初にプレイヤーカードを 2 枚引く「手札の補充」を行い、ターンの終わりには感染カードを数枚引く、「感染フェーズ」を行わなければなりません。ここで一つ触れておきたい事は、「手札の補充」で引く、プレイヤーカードの中にはエピデミックカードというハズレカードのようなものが数枚入っているということです。このカードを引いてしまうと特殊な処理を行わなければならなくなるのですが、詳しいことは省略します。**とりあえずヤバい**カードだと思ってもらえば問題ないと思います。

ゲームを始める時、初期配置で 9 つの都市に感染者が発生した状態になります。うち 3 つの都市は最初から 3 つの病原体コマが積まれている状態です。最初からクライマックスとはこの事です。

プレイヤーは新型ウイルス対策チームの一員となります。対策チームには科学者や研究員、衛生兵などの役職がいくつかあって、それぞれ特殊能力を持っています。プレイヤーは自分の役職の能力を活かしながら、世界地図を模したマップの上でウイルスの火消しをしつつ治療薬を完成させるために奔走します。

書くことはまだまだあるのですが、これ以上長くても誰も読まない仕方がないと思うので省略して、ゲームをする上で一番大事な勝利条件、敗北条件だけを並べます！

- 治療薬を 4 つ作ると**勝ち**。世界は救われる！
- アウトブレイクが 8 回起こると**負け**。人類滅亡。
- 病原体コマをおかないといけないのに、コマがないと**負け**。人類滅亡。
- プレイヤーカードを引く時に、山札にカードがないと**負け**。人類滅亡。

勝利条件が 1 つに対して敗北条件が 3 つもあるので、割と難しいゲームなのでは？と思った方もいるでしょう。**その通りです**。いえ、少し訂正します。正しくは「かなり難しくすることもできる」ゲームです。例えば WORD 編集部では、正規のプレイ人数である 2 ~ 4 人を超える、役職の数的限界の 5 人でプレイする場合が稀に良くあるので、カードは分散して揃わない (=治療薬が作れない)、山札はどんどんなくなる、という高難度プレイが常態化しています。(もちろんエピデミックカードはフル投入 (6 枚)) このレベルにすると、僕はやっていないのですがどうやらプレイヤー全員に 1 ターン目が訪れる間も無く人類が滅亡することもあるらしく、かなりスピーディー () なゲームになるようです。

ルールについては書くことが結構多くて正直やってられない大変なのでちょっと簡略化してしまいましたが、ここまで読んでなお興味が尽きない人は Wikipedia などを参照するか、ホビージャパンから販売されている商品を購入して実際にプレイしてみてください！

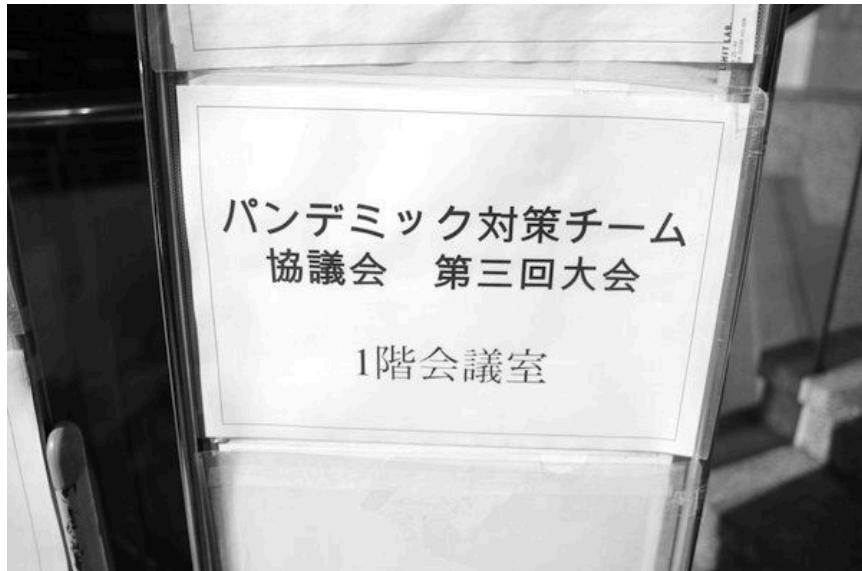


パンデミックのプレイ風景

パンデミック大会に行ってきたよ

先日渋谷にある日本薬学会長井記念館で開催された「第三回パンデミック大会」に我々 WORD 編集部の精銳 2 人 + 初心者 1 人（僕）も参加してきました。

この大会は毎回実際のウイルス関連の出来事を記念して開催されていて、今回は「人に感染するほぼすべての A 型インフルエンザの感染を阻止する抗体の発見」を（勝手に）記念して開催されました。



会場では他に厳かな勉強会が行われていたようだがこちらも負けていない

日本薬学会という、会場のガチっぽい雰囲気に身の引きしまる思いをしつつ、会場入りしました。



WORD といえば MAX コーヒー（後ろの紙には WORD 編集部と書いてある）

ぱんでみっく!!

■大会ルールの説明

今回の大会のルールをさらっと説明します。

- 主催者があらかじめ決めた順に並べられたカードを全員が使用し、同じ条件下でいかに早くクリアできるかを競う。
- 難易度ごとに各1回、合計4回のゲームを行った合計ポイント数で順位を決める。
- ポイントは下の表の通りで、クリアで満点、ゲームオーバーになった場合も作った治療薬1つにつきポイントが加算される。
- 4回中1回だけゲーム開始前に「大勝利宣言」^{*3}をすることができ、宣言をした回のゲームをクリアするとボーナスポイントが加算される。

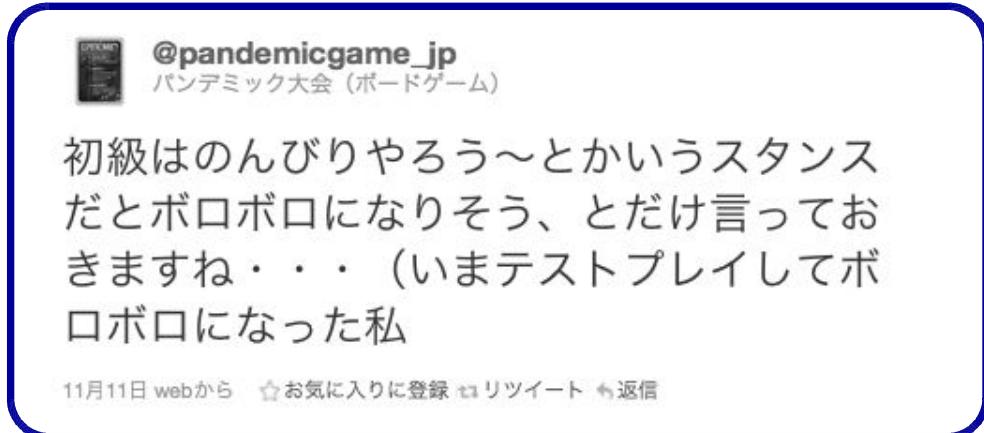
第1試合	初級（エピデミックカード：4枚）	時間内クリア：10点 大勝利宣言ありでクリア：+5点 タイムアップ/失敗：作成済み治療薬1つにつき2点
第2試合	中級（エピデミックカード：5枚）	時間内クリア：20点 大勝利宣言ありでクリア：+10点 タイムアップ/失敗：作成済み治療薬1つにつき4点
第3試合	上級（エピデミックカード：6枚）	時間内クリア：30点 大勝利宣言ありでクリア：+15点 タイムアップ/失敗：作成済み治療薬1つにつき5点
第4試合	伝説級（エピデミックカード：7枚）	時間内クリア：40点 大勝利宣言ありでクリア：+20点 タイムアップ/失敗：作成済み治療薬1つにつき8点

前もって用意された山札に沿ってゲームが進むので、主催者との駆け引きも重要なポイントです。そしてなにより、人類を救ってやるというアツい気持ちが大事です！！公式サイトの持ち物にもそんなことが書いてありました！！！

*3 スーパーひとし君人形のようなもの

■第1試合

まずは初級の第1試合ですが、大会公式Twitterアカウントでの



という発言を受け、難易度は初級ではありますが慎重にプレイしました。

しかしながら、こちらはWORD編集部での高難度プレイを積み重ねた精銳、割とあっさりクリアした後、「手札にカードがすぐ集まって楽」「山札が減るの遅くて良い」などという感想を漏らしていました。きっと人類も「あー確かにちょっとやばかったカナ~」くらいのダメージで済んだ事でしょう。

順位の方も、慎重に、とは言っても結構急ぎ気味にやったつもりだったのでクリア直後は僕たちの順位は2位だと思っていました（直前に1チーム先にクリアしていたのは見たので）。ところが流石は対外試合、実はとんでもない速さでクリアしていたチームがもう1チームいたということがわかりました。この事実にチビる程ビビってしまった僕は、不手際でわずか2本しか持ってくることができなかつたMAXコーヒーの内の、貴重な1本目をキメざるを得ませんでした。ということでWORD編集部は全9チーム中の3位スタート、他のチームのレベルの高さを感じさせる緒戦になりました。

■第2試合

続く第2試合、エピデミックカードが1枚追加されレベルが上がります。きっと「まあ1枚だしあんまり変わらないでしょ～～（ホジホジ」とお思いの人も多いかと思いますが、4枚と5枚の差は実は結構大きいです。と言うのも、エピデミックが発生すると感染率⁴が上昇するのですが、この上昇の仕方は【2→2→2→3→3→4→4】となっているため4回までは感染フェーズに引くカードの枚数は最大でも3枚なのですが、5枚になると最大4枚に増えるのです。感染フェーズは基本的に毎ターン行うのでこれは割と辛くて、どう辛いかというと、スゴく辛いんです！！

僕たちの試合の内容ですが、予想以上にアツい展開になりました。エピデミック後の引きの悪さが出て、アウトブレイクの回数が嵩んでしまった上に、世界各地に3つの感染コマが積まれた地域が散在している状態でまさに一触即発の事態まで陥ってしまったのです（絶望）。治療薬を作り終わるのが先か、アウトブレイク回数がMAXに至って世界が滅ぶかどっちだ！？！？という状況にな

*4 ターンの終わりに引く感染カードの枚数が決まる。

ぱんでみっく!!

ったのですが、マンガばかりの先の読み合いの末に辛くもクリアすることができました。ここまでウイルスにやられていてはきっと人類も半滅亡状態でしょうが、とりあえず世界を救う事ができました！激アツな試合を制した喜びを感じつつ、僕は勝利の美 MAX コーヒーをキメました（2本中の2本目なので後のこととも考えて少しだけ）。

やっぱり結構難しかったのか、クリアできたチームは WORD 編集部の他は 1 チームしかいなかつたのでクリア順位は 1 位でした！！

■第3試合

「クリアできそうな最高難度ではないだろうか」と話し合っていた上級の第3試合、ここで WORD 編集部チームは大勝利宣言をしました。大勝利宣言をした以上、この試合での敗北は大敗北を意味すると言っても過言では無いでしょう。前の試合で、「主催者は誰にどの色の治療薬を作らせようとしているのか」という意図も考慮していく必要があると気づかされたので、序盤からその辺りも含めて主催者の意図を汲み取ろうゲームっぽい感じでいきました。小中高校での国語教育で作者の意図読み取り能力を培ってきた僕たちですから、その辺の読みは結構得意なのです。読みが割と当たったおかげか、第2試合よりもむしろ楽ちんな感じにクリアする事ができました。人類も安泰です。

クリアチームの数も多く、第3試合は横並びな感じでした。クリア順位は2位でした。

■第4試合

いよいよ最終戦。エピデミックカードは最大の 6 枚を超えて 7 枚になります！当然市販のパンデミックには 7 枚目のエピデミックカードは入っていないので、主催者のトミールさんがホビージャパンさんに「エピデミックカードあるだけください！(追真)」とお願いして入手してくださったカードを使用しました。

ラスボス戦ということで、僕も残り少なくなった MAX コーヒーをしっかりとキメて気合を入れて試合に臨みました。最終戦は時間制限は無し（のようなもの）で行われるということで、順位は度外視してとにかく人類を滅ぼさないことを考えながらじっくり進めていく作戦をとりました。

しばらくすると、続々と人類を滅ぼさせてしまったチームが現れだし、終了したチームの緊張感が解けた感じと、ゲームが続いているチームの緊張感が混在する不思議空間になってきました。僕たちはこれまでの 3 試合を教師データとして学習し、さらに強化された主催者の意図読み取り能力と体得したパンデミックの定石を駆使し、できるだけ被害を抑えつつ治療薬を作っていました。

それからもうしばらくすると、気づけば WORD 編集部ともう 1 チーム以外の全チームが敗北しているという状況になっていました。ゲームを終えたチームの人たちはひとしきり歓談を終えると残りのチームの観戦を始め、僕たちの周りにも人垣ができていました。ゲームセンターで人垣に囲ま

れて超人プレイしてゐる人の気持ちが少しあわかったような気になりつつ、さらに慎重かつ正確なプレイを進めていき、無事伝説級をクリアする事ができました！！人類の歴史に新たな伝説が作られた瞬間でした！！

結局もう1チームの方も僕たちが終わってからしばらくして、僕たちの第2試合目のようなハラハラドキドキ展開を制して、見事伝説を作っていたので、計2チームが伝説級をクリアする事ができました。僕は勝利の美MAXコーヒー（ちょっと余ってた）を飲み干した後、もう1チームのゲームを観戦していました。



約束された勝利の MAX コーヒー

■結果発表

ここまで飛ばさずに読んで下さった、すばらしい方々はすでに予想はできていると思いますが、我らがWORD編集部チームは筑波大学代表を勝手に表明しても差し支えない程の活躍を見せ、全難

	2.音素 子音 複数形 本部	3. 音 子音 複数形 地圖用字	9. WORD 複数形	WORD 複数形	0. 水化合物	17 FHF	8. 单数 複数形	9. カタキン	10. 人間性
初級	1	5	X(3)	3	6	7	X(3)	2	4
	10	10	6	10	10	10	6	10	10
中級	(2)	(2)	(2)	①	(2)	8	(3)	(2)	(3)
	8	20	8	20	8	12	8	12	(2) 8
上級	④	(3)	(3)	③	①	⑦	⑤	⑥	② 30
	30	15	15	45	30	30	45	85	30
伝説	(1) 8	(2) 40	(2) 16	① 40	(2) 16	(3) 24	(1) 8	(3) 24	(1) 8
合計	56	85	45	115	64	96	69	291	56

スコアボード

ぱんでみっく!!

ななんなんと、上位入賞チームには賞品が用意されていました！ウレチー！

賞品には、すごい尖って痛いけどイカしてるトロフィーと、協賛のホビージャパンさんからテーブルゲーム⁵を戴きました（パンデミックではありませんでした）。それに加えて、映画「コンティジョン」⁶の鑑賞券を2枚戴きました！映画は好きなのでとても嬉しいです。だが少し待ってほしい……チームは3人組なのです！！

「はーい2人組作ってー」

この台詞が思わず脳裏をよぎりましたよ。完全に狙ってますねコレは……！



栄光を勝ち取った

普段テーブルゲームをあまりやらない僕ですが、今回の大会では非常に楽しい時間を過ごすことができました。主催者のトミールさんや協賛のホビージャパンさん、どうもありがとうございました！！

さいごに

パンデミック、面白いので1回やってみてください！！！映画、終わってしまう前に観に行かねば！！

*5 パンデミックではなかった。

*6 実写版パンデミックのような映画らしい。

エターナル餃子チケットのすすめ

文 編集部 IX

■ はじめに

最近、私は学食^{*1}である声を聞きました。「学食飽きた」という声を。連日利用していると飽きが来てしまうのか……空いた小腹を埋めにカレーうどんこぶた丼セット^{*2}を食べに行ったり、おやつ用のチキンカツ弁当^{*3}を買いに行くぐらいでしか利用していない私には寝耳に水でした。

よし、ならあそこのこと紹介しようじゃないか。学食価格に100円程度上乗せすると豊かになれるに定評のあるあのフードコートを……。

■ あのフードコートとはいったい……

つくばクレオスクエア内にあるフードコートです。つくばセンターから徒歩数分のサンクスの横の入り口から入ってまっすぐ行ったところにあります。

九つの飲食店がそのエリアに含まれており、360の座席とテーブルがあります。web上^{*4}に公開されている各店舗の紹介は以下のようになります。

■ 店舗紹介

・香辛飯屋

カレー専門店ならではの手づくりカレーは、野菜と果物を秘伝のスパイスと一緒にじっくり煮込んだ欧風カレー。

季節のカレーも是非ご賞味下さい。テイクアウトも承ります。

・築地 銀だこ

外はカリッ。中はトロッ。たこはプリッとした食感が自慢の江戸前たこ焼き。期間限定のたこ焼きもどうぞ。

・モスバーガーファクトリー

安心・安全・健康そしておいしさをキーワードに商品提供を心掛けています。
ご来店お待ちしております。

*1 学食：学生食堂の略。自炊しない方の学生の生命線と言える。周辺の飲食店と比べると割安で済む。

*2 カレーうどんこぶた丼セット：パーティをやる方の学食で購入できるセットメニュー。体育後のエネルギー補給に最適。

*3 チキンカツ弁当：パーティやらない方の学食で購入できる弁当。注文を受けてから容器に詰めるため、半額にはならない。オオカミも来ない。平和ダナー。

*4 web上：http://www.creo-sq.com/sc_manage/search_shop.php

油淋鶏セット+焼き餃子 = 590円

・ベリーベリーカフェ

こだわり素材を使った本格的なクレープは数十種類から選べます。各種オリジナルドリンクや季節のフレーバーが楽しめるソフトクリーム、ジェラード、フラッペなどデザートの他、しっかり食べられるパスタメニューも充実。

・ステファングリル

新しくメニューも増え、価格もお求めやすい値段になりました！

メインメニューとセットでお子様メニューが280円になります！

ふわふわ卵のオムライスとジューシーな手作りハンバーグをどうぞご賞味ください！

・釜揚げ讃岐うどん 丸亀製麺

打ちたて、茹でたて、使う小麦粉は厳選した国産小麦粉。常に本物のうどんの美味しさをお伝え中です。天ぷら、おむすびにもとことんこだわり、うどんに負けない美味しさをご用意しておりますので、是非お好みにあわせてご自由にお選び下さい。

・サブウェイ

店内で焼き上げたパンにハムやチキン、6種類の野菜を挟んだフレッシュサンドイッチ。お客様の目の前で作るから、野菜の好き嫌いやドレッシングの変更などのわがままもOK。いつも作りたてのサンドイッチを提供いたします。

・ちやぶ屋とんこつらあ麺 CHABUTON

テレビチャンピオン「新行列店ラーメン職人選手権」優勝「ちやぶ屋」店主森住康二氏が手掛けるプレミアムラーメン店。

「ミシュランガイド（ロサンゼルス版）」でも紹介された世界が認めた腕前をどうぞご堪能下さい。

・王記厨房

出来立ての熱々な本格中華料理をお得なセットでお客様にお届け！

ご注文が入ってから鍋をふり、いつも熱々の料理はもとより、日本人に人気の麺類や、馴染みのある一品料理を定食やセットにしてお求めやすい価格で、お客様に本格中華料理を提供します。

**ちょっと無難すぎますね。
私の主観を交えた紹介で
白黒はっきりさせていきましょう……。**

■ わかりやすい店舗紹介

・香辛飯屋

本店の方で有名な某チャレンジメニュー^{*5}は実施されていない。カレーは確かにうまいが学食と比べると割高。

ただし、結構長い間キャンペーンでやっているササミカツカレー(500円)は例外。ワンコインで済む上、カツのサクサク感が段違い。

また、web 上でのアイコンがカレーパンを掲げている画像だが、カレーパン売ってるイメージはありません。

・築地 銀だこ

一店舗で食事した後にもう一品行きたい、そんなときに重宝するお店。

8がつく日にスタンプカードのスタンプ数が二倍になる。スタンプカードは、「赤→銀→金」の順にステップアップするが、私はまだ赤カード。どうみてもにわかです 本当にありがとうございます いました。

期間限定のたこ焼き^{*6}は、100円増しになるため手が出しづらいが、日によっては50円増しで普通のたこ焼きとのハーフ&ハーフを提供している様子。

先日、私は見事にホイホイされた。食べる醤油おいしい。

・モスバーガーファクトリー

ファクトリーとはいったい……。

幼少期には家族と一緒に長泉店に頻繁にいったなあ。セットを頼まずに、バーガーばっかり三個くらい食ってた記憶がよみがえる。食べ盛り故致し方なし。

・ベリーベリーカフェ

おしゃれなカフェ的な雰囲気のお店。私は余り近寄ったことはない。クレープって腹にたまらないのに高いじゃん？

・ステファンギリル

あの鉄板はどうみても、ヘ○ッハ○ーランチ。以前、ケチャップオムライス半額というとんでもないイベントをやっていたのを発見し、利用しようとしたが、すでに「かけうどん+かしわ天+野菜かき揚げ+大量の青ネギ+多めに掬って後悔した天かす+たこやき八個」を食していたため保留。それっきり。

後輩曰く、安くお肉が食べられるいい所とのこと。今度利用しよう、うどん食ってから。

*5 某チャレンジメニュー：1.5kg挑戦カレーのこと。20分以内に完食すると無料。小食の筆者は失敗時の罰金が怖くて挑戦していません。

*6 期間限定のたこ焼き：現在は「ぶっかけネギ醤油」。過去には、卵とフライドベーコンが乗った「チーズオムたこ」、夏限定の「涼風とろろ天つゆ」などが存在している

油淋鶏セット+焼き餃子= 590 円

・釜揚げ讃岐うどん 丸亀製麺

昼時はめちゃくちゃ混む。主婦に学生、サラリーマンと客層は幅広い。うどんをお盆に乗せてから会計をしに行くまでの間に、誘惑の天ぷらロードが広がっており、私はよくホイホイされてしまう。かしわ天⁷にサツマイモ天、野菜かき揚げ、小海老のかき揚げ等々。うどんだけ食べると安上がり、天ぷらをつけると知らず知らずのうちの安上がりでない、そんなお店。

天かす、青ネギの入れ放題は実にうれしい。

【例 1】

かけうどん(並)+青ネギ+天かす+ショウガ+ごま

【例 2】

肉ゴボウぶっかけうどん(大)+かしわ天+かしわ天+野菜かき揚げ+青ネギ

例 1 が安上がりで済む構成、例 2 がいつの間にか高くなっていた、とある私のおやつ構成となっています。

出汁にカレーをかけるタイプのカレーうどんを食べることが出来る。青ネギをたっぷりと添えて食していただきたい。出汁にカレーを溶いて一体化させるタイプのカレーうどんは ZEYO⁸ で食べることが出来る。

・サブウェイ

オーダーメイドサンドウィッチ専門店。アメリカ発祥とは思えないくらい野菜が摂取できるお店。注文の仕方が複雑そうだから敬遠しているという人が多いらしいですが、そんな心配要りません！

チーズローストチキン、ウィートのフットロング、オリーブ抜きレタスピクルス目一杯増量おすすめドレッシングでお願いします

この呪文を覚えるだけで美味しく食事が出来ます（注文に迷ったら、「おまかせで」と申し出ればいいとのこと）。

・ちゃんことんこつらあ麺 CHABUTON

なんかチャンピオンらしいけど、混んでるところはあまり見たことない。きっと回転率がべらぼうにいいのだろう。ラーメン一杯はちょっと高めだが、替え玉は結構安い。

*7 かしわ天：鶏肉の天ぷら。脂身がほどよく食べ応えあり。100 円。

*8ZEYO：カレーうどん専門店 ZEYO のこと。平砂側の筑波大学の出口付近にある。学生向けメニューの価格が崩壊している。

• 王記廚房

おわかりいただけただろうか？

油淋鶏(ユーリンチー)セットがうまいのである。揚げたての唐揚げに刻みネギが添えられ甘くない甘酢ソースがかけられた物が五つ、白飯、ザーサイ、スープがついて 590 円。

囁いた瞬間あふれる肉汁、香ばしい皮、外はサクッ中はジューシーと王道を行く唐揚げをいただくことが出来ます。

さらに、セットメニューを頼むと次回から使える無料券がついてくるのです。

その内容は、

ウーロン茶、コーラ、メロンソーダ オレンジジュース、焼き餃子(5ヶ)

の中から選んだ一品を無料で提供してもらえるというもの。

なんと、飲み物以外に焼き餃子が選択肢にあるではありませんか！

しかも、この無料券はセットメニューを頼むと必ずついてくるのです。つまり、一度この券を手に入れてしまえば、食事ごとに焼き餃子を無料で半永久的に追加し続けることが出来るのです。

私は、これを**エターナル餃子チケット**と名付け、財布の札を入れる場所に入れて常時携帯しているのである。

■ まとめ

このように少し移動時間がかかりますが、つくばには魅力的な食事処がたくさんあります。学食に飽きたというそこのあなた、移動するのがめんどくさいとななど言わず、新境地開拓に出かけてみませんか？

画像が全くなくてわかりにくい？ 腹減ったなら、自分で行ってその目で 確かみてみろ！^{*9}

*9 確かみてみろ！：今は無きアーケードゲーム専門誌、ゲーメストであった伝説の誤植の一つ。

電子の歌姫はアイドルの夢を見るか -extend-

文 編集部 Genyakun

はじめに

今年の4月末に、「初音ミク -Project DIVA- Ver2.5」（現：初音ミク -Project DIVA- extend）というゲームが突如発表されました。

この記事では、その「初音ミク -Project DIVA- extend」が11月10日に発売されたことを記念した攻略記事となっております。

そのため、ネタバレを嫌う方や、音ゲーなんて嫌いというような方はここから数ページをホチキスやノリで固定するか、記事の画像だけ見てください。

初音ミク -Project DIVA- extendとは

初音ミク -Project DIVA- extend(以下、DIVA extend)とは、SEGAが開発したPSPで動作するリズムアクションゲーム(つまりは音ゲー)です。初音ミク -Project DIVA- はシリーズ物となっていて、DIVA extendが三作品目に当たります。過去の作品の紹介につきましては、筆者が過去に書いた記事^{*1}をご参照ください。

では、具体的にどんなゲームかと言いますと、画面上に配置されるマーカーに応じたボタンを押すだけという簡単なゲームです。難易度については、EASY, NORMAL, HARD, EXTREMEの四段階^{*2}となっており、初めての方もEASYモードで安心してプレイすることができます。また、楽曲数も前作(初音ミク -Project DIVA- 2nd)と比較すると10曲程度減って36曲になったものの、36曲中19曲は新規収録曲のため、前作をやりこんだユーザも安心(?)してプレイすることができます。

また、前作からあったPV鑑賞機能や、自作譜面やPVを作るツールなどもブラッシュアップして収録されています。

*1 WORD15号「非実在WORD編集部号」より「電子の歌姫はアイドルの夢を見るか」とWORD16号「侵略されたいでゲソ号」より、「電子の歌姫はアイドルの夢を見るか -APPEND TRACK-」の二本。次のWebサイトより閲覧が可能 <http://www.word-ac.net/>

*2 楽曲ごとにHARDを出すにはNORMALをクリアすることが必要で、EXTREMEを出すのにはHARDをクリアすることが必要

Project DIVA 最新作攻略

画面説明

まずは実際のプレイ画面を見てみましょう。たとえば、買って最初にプレイできる楽曲の一つである「SPiCa -39's Giving Day Edition-」の EASY 譜面はこのようになっています。



EASY で使用するのは○キーだけで、基本的には同時押し³と、この画面にはない長押し⁴の練習がメインです。また、後述する縛りプレイや、一部の高難易度曲をプレイする場合、左下のソングエナジーゲージ（以下エナジーゲージ）が割と大切になってきます。エナジーゲージはコンボをつなげているといつの間にか増えて、ミスをすると減っていきます。無くなるとその時点で曲が中断して、MISS × TAKE 扱いになってしまいます。また、画像では見えにくいですがエナジーゲージの外側にあるゲージは現時点での成績が暫定的に表示されています。前作とは異なり、後半になって突然モリモリ増え出すと言うことはあまりなくなったようです。

基本的な攻略方法

まず基本的な攻略方法についてですが、大体次のようなことを覚えておくと良いでしょう。

・説明書を読む

まず説明書を読みましょう。この記事には書いてない大切なことがいっぱい書いてあります。

・原曲を聞き込む

大体の譜面（EXTREME 譜面以外）は歌詞がそのまま譜面になっているので原曲を一度聞いておくと楽でしょう。

特に今作ではオリジナルサウンドトラック⁵がちゃんと発売されているのでそれを買って聞きましょう！

*3 「←」のような形をしたマーカで、基本的に○×△□と同じ方向の矢印ボタンを押す。例えば「←」なら○と右矢印ボタンを同時に押す

*4 普通のマーカが長く伸びてるやつ。初めのマーカで押し始め、終わりのマーカでボタンを離す

*5 初音ミク-Project DIVA-extend Complete Collection <http://www.amazon.co.jp/dp/B005J7CSCW>

・マーカをちゃんと確認する

個人的な感覚ですがタイミング判定が早押しについては特に厳しい⁶ので注意しましょう。逆に遅く押す分には判定が緩いのでマーカとターゲットマーカが重なったのを確認したくらいで押してもコンボがつながります。

・同時押しの連打は片方のキーを押しっぱなしで対処可能

同じキーの同時押しのマーカが連続している場合には、片方は入力しっぱなしでも問題なく判定されます。たとえば、「→→→」のように→型の同時押しのマーカが連続して来た場合は矢印キーを押しっぱなしで○キーだけをマーカの数だけ押せば⁷処理が可能です。これの応用で同時押しに挟まってる別キーのマーカに対しても同じような事⁸をすることが可能です。

・同じキーの連打は左右に負荷分散が可能

たとえば「××××」というマーカがあったとしたら、「×↓×↓」と交互に入力することで片方の指への負荷を下げる事が出来ます。しかし、交互に入力しているつもりがタイミングがずれてコンボが切れる事があったり、そもそも HARD までの大体の曲は親指で対処可能な範囲内の譜面なので最初のうちから習得する必要はありません。

・その他

タイミング判定で SAFE 以下⁹を取るとコンボが途切れてしまうので注意しましょう。また、個人差はありますがボタン音を切ってリズムを聴き取りやすくするのも一つの手です。

成績について

肝心の成績については総ノート数とタイミング判定で COOL か FINE だった数の割合(以下 C/F 率)で決定し、悪い順に MISS × TAKE(プレイ中に終了)→ CHEAP(C/F 率 84%以下)→ STANDARD (C/F 率 85%) → GREAT (C/F 率 95%) → EXCELLENT (C/F 率 97%) → PERFECT (C/F 率 100%) というランク分けになっています。

また、スコアは成績評価には関係ないのですが、ハイスクアを目指すためにはコンボ数や、チャンスタイム(マーカが虹色の部分)でのミスを限りなく無くす(言うならばその間だけでもコンボがつながるようにする)事が重要です。

これらのことを見ておけば多くの人は EASY を一周する頃には全曲がプレイ出来るようになっているはず¹⁰なので、その後は次の難易度を一周ずつしていくばすべての難易度が解放されると思います¹¹。ちなみに、EXTREME 譜面の入門には難易度★6 の「Starduster」や「歌に形はないけれど」、★7 の「Yellow」あたりが良いでしょう。

*6 焦れば焦るほどコンボが切れるので連打も気持ち遅めにした方が良かつたりする

*7 この場合 3 回

*8 「→□→」のノートがあったら右矢印キーを押しっぱなしで、「○□○」と入力すれば処理が可能。これをさらに応用すると EASY 譜面でイントロから右矢印押しっぱなしで○キーだけ操作できるため、PERFECT トライアル時のミスを減らすことが出来る

*9 ボタンを押したときのタイミング評価は悪い順に WORST → SAD → SAFE → FINE → COOL くなっている

*10 最初は 4 曲しか見えないが、出てきている曲をクリアしていくばどんどん解放されていく

*11 EASY, NORMAL, HARD を各一周すればアイテムや衣装はほとんど解放されるが、後述の通り解放されただけでは使えない

Project DIVA 最新作攻略

引き継ぎ要素について

もし、あなたが 2nd をプレイしている場合、各セーブデータにつき一度だけプレイデータの引き継ぎを行うことができます。プレイデータを引き継いだ場合に、引き継がれるデータは以下の通りです。

- ・DIVA ポイント^{*12} + 10,000DIVA ポイント
- ・ルームアイテムの配置
- ・購入したモジュール（衣装）・ルームテーマ・ルームアイテム
- ・プレイレコードの一部（総 DIVA ポイント、称号の一部）
- ・登録済みのフレンド（50 名まで）

結局、何が言いたいかと言うと、「2nd をプレイし尽くした人ほどお得な設計」になっています。

たとえば、2nd を遊び尽くして DIVA ポイントが 9,999,999（カンスト）になってる人は DIVA ポイントもそのまま引き継がれ、衣装も引き継がれます。つまり、後述する DIVA extend で新規に購入する衣装の DIVA ポイントを考えても余りある量が何もせずに手に入るわけです。これは引き継がない手はないでしょう。

何よりも、extend より 2nd の方が簡単で楽しいと筆者は思ってるくらいですから…

アイテム要素（縛りプレイ）について

引き継ぎ要素の説明をしたところで、チャレンジアイテムと、ヘルプアイテムについて説明します。

本作のアイテムには以下の物があり、次のような効果とポイントがあります。

アイテム名	効果	ポイント
プレイアシスト	SAD,WORSTがSAFEになるが、リザルトがCHEAPになる	1,000
コンボガード	10回までSAFE、SADがFINEになる	1,500
リカバリー	ソングエナジーが0になった時にエナジーを全回復	2,000
COOL&FINE	COOL, FINE以外の判定をWORSTにするが、獲得DIVAポイントが2倍になる	3,000
サバイバル	ソングエナジーが増加せず、減少量が増加するが、獲得DIVAポイントが4倍になる	5,000

下二つが縛りプレイ向けのアイテムで、特に熟練した方にお勧めなのが「サバイバル」アイテムです。これは WORST を 3 回程度を出すと即 MISS × TAKE になるものの、成功すれば 4 倍の DIVA ポイントが獲得できるということで、いつも PERFECT に近いところまで出している得意な楽曲や難易度で挑戦してみるといいでしょう。

*12 アイテムや、モジュール（衣装）を購入する際に必要になるゲーム内ポイントのこと。ポイントは楽曲をクリアすると獲得できる

収録楽曲の解放順序について

プレイを始めた段階では、チュートリアル以外に「ねこみみスイッチ」、「孤独の果て」、「Palette」、「SPiCa -39's Giving Day Edition-」をプレイすることができます。これら 4 楽曲を起点として、楽曲をクリアするたびに別の楽曲が解放されます。以下に解放される条件と収録曲を示します。

楽曲名	解放条件
ねこみみスイッチ	初期楽曲
孤独の果て	初期楽曲
Palette	初期楽曲
SPiCa -39's Giving Day Edition-	初期楽曲
Starduster	「ねこみみスイッチ」クリア
那由他の彼方まで	「孤独の果て」クリア
星屑ユートピア	「Palette」クリア
タイムリミット	「SPiCa -39's Giving Day Edition-」クリア
歌に形はないけれど	「Starduster」クリア
炉心融解	「那由他の彼方まで」クリア
えれくとりっく・えんじえう	「星屑ユートピア」クリア
Yellow	「タイムリミット」クリア
千年の独奏歌(DIVA edit)	「歌に形はないけれど」クリア
右肩の蝶 -39's Giving Day Edition-	「炉心融解」クリア
あなたの歌姫	「えれくとりっく・えんじえう」クリア
パズル	「Yellow」クリア
忘却心中	「千年の独奏歌(DIVA edit)」クリア
リンリンシグナル -Append Mix-	「右肩の蝶 -39's Giving Day Edition-」クリア
Just be Friends	「あなたの歌姫」クリア
melody…	「パズル」クリア
番凧	「忘却心中」クリア
カラフル×メロディ	「リンリンシグナル -Append Mix-」クリア
ルカルカ★ナイトフィーバー	「Just be Friends」クリア
StargazeR	「melody…」クリア
金の聖夜霜雪に朽ちて	「番凧」クリア
いろは唄	「カラフル×メロディ」クリア
*ハロー、プラネット。(I.M. PLSE-EDIT)	「ルカルカ★ナイトフィーバー」クリア
結んで開いて羅刹と骸	「StargazeR」クリア
初音ミクの消失	「金の聖夜霜雪に朽ちて」クリア
パラジクロロベンゼン	「いろは唄」クリア
裏表ラバーズ	「*ハロー、プラネット。(I.M. PLSE-EDIT)」クリア
こっち向いてBaby	「結んで開いて羅刹と骸」クリア
初音ミクの激唱	「初音ミクの消失」クリア
カラフル×セクシィ	「パラジクロロベンゼン」クリア
ローリングガール	「裏表ラバーズ」クリア
積乱雲グラフィティ	「こっち向いてBaby」クリア

Project DIVA 最新作攻略

モジュール（衣装）について

DIVA extend では、キャラクタや衣装を選ぶことができ、衣装のことを「モジュール」と呼んでいます。また、DIVA extend には従来の初代 DIVA、DIVA 2nd のモジュールに加えて、新規に追加されたモジュールが存在します。これらは、曲をクリアするごとに解放され、ショップにて DIVA ポイントを使って購入することができます。そこで、ここでは参考として DIVA extend で新規に収録されたモジュールの一覧と、取得条件や必要な DIVA ポイントを以下に示します。

キャラクタ	モジュール名	取得条件	難易度	ポイント
初音ミク	ソニックスタイル	「ねこみみスイッチ」クリア	NORMAL	20,000
初音ミク	オービット	「SPiCa -39's Giving Day Edition-」クリア	NORMAL	20,000
初音ミク	ホワイト・イヴ	「Starduster」クリア	NORMAL	20,000
初音ミク	レーシングミク2010ver	「タイムリミット」クリア	NORMAL	20,000
初音ミク	チロル	「歌に形はないけれど」クリア	NORMAL	20,000
初音ミク	パッチワーク	「パズル」クリア	NORMAL	20,000
初音ミク	フェイ・イエン スタイル	「melody…」クリア	NORMAL	20,000
初音ミク	Hello world.	「*ハロー、プラネット。(I.M. PLSE-EDIT)」クリア	NORMAL	20,000
初音ミク	ラセットムクロ	「結んで開いて羅刹と骸」クリア	NORMAL	20,000
初音ミク	コンフリクト	「裏表ラバーズ」クリア	NORMAL	20,000
初音ミク	回転少女	「ローリングガール」クリア	NORMAL	20,000
初音ミク	シャイニー	「積乱雲グラフィティ」クリア	NORMAL	20,000
初音ミク	春香Style	「えれくとりっく・えんじえう」クリア	NORMAL	20,000
初音ミク	初音ミク クリスマス	「金の聖夜霜雪に朽ちて」クリア	NORMAL	20,000
初音ミク	TYPE2020	「カラフル×メロディ」クリア	HARD	20,000
初音ミク	初音ミク アpend	「Yellow」クリア	HARD	20,000
初音ミク	雪ミク2010	NORMAL全曲クリア	NORMAL	20,000
初音ミク	雪ミク2011	HARD全曲クリア	HARD	20,000
初音ミク	レーシングミク2011ver.	難易度EXTREMEでどれか一曲クリア or 初音ミクで100回プレイ		20,000
鏡音リン	鏡音リン アpend	「リンリンシグナル -Append Mix-」クリア	NORMAL	20,000
鏡音リン	ブラックスター	「孤独の果て」クリア	NORMAL	20,000
鏡音リン	鏡音リン 陽炎	「いろは唄」クリア	NORMAL	20,000
鏡音リン	鏡音リン スクールウエア	「右肩の蝶 -39's Giving Day Edition-」クリア	NORMAL	20,000
鏡音リン	鏡音リン クリスマス	「金の聖夜霜雪に朽ちて」クリア	NORMAL	20,000
鏡音リン	ネームレス No.1	「炉心融解」クリア	HARD	20,000
鏡音リン	鏡音リン 蘇芳	「リンリンシグナル -Append Mix-」クリア	HARD	20,000
鏡音リン	レーシングリン2010ver.	「いろは唄」クリア	HARD	20,000
鏡音リン	亜美・真美Style	「えれくとりっく・えんじえう」クリア	HARD	20,000

次のページへ続く

キャラクタ	モジュール名	取得条件	難易度	ポイント
鏡音レン	鏡音レン アペンド	「リンリンシグナル -Append Mix-」クリア	NORMAL	20,000
鏡音レン	ブルームーン	「那由他の彼方まで」クリア	NORMAL	20,000
鏡音レン	鏡音レン スクールウエア	「右肩の蝶 -39's Giving Day Edition-」クリア	NORMAL	20,000
鏡音レン	鏡音レン クリスマス	「金の聖夜霜雪に朽ちて」クリア	NORMAL	20,000
鏡音レン	ストレンジダーク	「パラジクロロベンゼン」クリア	NORMAL	20,000
鏡音レン	鏡音レン 藍鉄	「リンリンシグナル -Append Mix-」クリア	HARD	20,000
鏡音レン	ネームレス No.7	「パラジクロロベンゼン」クリア	HARD	20,000
巡音ルカ	サイレンス	「Palette」クリア	NORMAL	20,000
巡音ルカ	ナギサ・レプカ	「星屑ユートピア」クリア	NORMAL	20,000
巡音ルカ	ナギサ・レプカ AS	「星屑ユートピア」クリア	NORMAL	20,000
巡音ルカ	サイバーネイション	「ルカルカ★ナイトフィーバー」クリア	NORMAL	20,000
巡音ルカ	フェアリーマカロン	「カラフル×セクシィ」クリア	NORMAL	20,000
巡音ルカ	巡音ルカ クリスマス	「金の聖夜霜雪に朽ちて」クリア	NORMAL	20,000
巡音ルカ	レーシングルカ2010ver.	「ルカルカ★ナイトフィーバー」クリア	HARD	20,000
巡音ルカ	魔女っ娘Style	「Palette」クリア	HARD	20,000
巡音ルカ	千早Style	「Just be Friends」クリア	HARD	20,000
KAITO	スミレ	「千年の独奏歌(DIVA edit)」クリア	NORMAL	20,000
KAITO	時雨	「番凧」クリア	NORMAL	20,000
KAITO	VFニンジャ	「初音ミクの消失」クリア	NORMAL	20,000
KAITO	VFニンジャ AS	「初音ミクの消失」クリア	NORMAL	20,000
KAITO	ホワイトブレザー	「千年の独奏歌(DIVA edit)」クリア	HARD	20,000
KAITO	カイト クリスマス	「金の聖夜霜雪に朽ちて」クリア	HARD	20,000
MEIKO	ローレライ	「忘却心中」クリア	NORMAL	20,000
MEIKO	紅葉	「番凧」クリア	NORMAL	20,000
MEIKO	セクシープディング	「カラフル×セクシィ」クリア	NORMAL	20,000
MEIKO	レーシングメイコ2010ver.	「番凧」クリア	HARD	20,000
MEIKO	怪盗ブラックテール	「ねこみみスイッチ」クリア	HARD	20,000
MEIKO	大正浪漫	「忘却心中」クリア	HARD	20,000
MEIKO	メイコ クリスマス	「金の聖夜霜雪に朽ちて」クリア	HARD	20,000
亞北ネル	部活少女	「あなたの歌姫」クリア	NORMAL	20,000
弱音ハク	ゴシック・パープル	「StargazeR」クリア	NORMAL	20,000
咲音メイコ	ブラックワンピース NS	「初音ミクの消失」クリア	NORMAL	20,000
咲音メイコ	ノスタルジー	「こっち向いてBaby」クリア	HARD	20,000
	合計			1,220,000

Project DIVA 最新作攻略

楽曲別攻略

ここまで、おおまかに DIVA extend のシステムとその周辺について説明してきました。

ここからはいよいよ楽曲別の攻略に入りたいと思います。ここで紹介する楽曲は extend 新規収録曲の中でも筆者が独断と偏見で選んだ難易度の高い楽曲のみとし、難易度もゲーム内で表示されている物を使っています。難易度の★は最大で9つまでです。攻略の参考にどうぞ。

番 亂 (BPM 102)

序盤は楽な楽曲ですが、「この身 木の葉と 吹かれて行こう」の後の間奏部分に下図のような「○○○○××××△△△△」のような 16 分の連打が入っています。

ここまでエナジーゲージを貯めていて満タンになっていた場合、適当に対応するボタンを連打しておけば SAFE と WORST、ちょっとの FINE と COOL が出てエナジーゲージがなくなるギリギリ手前でチャンスタイムに入るはずです。チャンスタイム中はエナジーゲージが変動しないので安心してください。チャンスタイム後は比較的難しくないので、そのまま抜けましょう。



いろは唄 (BPM 172)

こちらも先ほどの「番夙」と同様に間奏部分に連打が入っています。

EASY	★★★
NORMAL	★★★★★
HARD	★★★★★★★
EXTREME	★★★★★★★★★

しかし、下図を見ていただくと分かりますが「□□ □□ ×× ×× □□」のように間が空いた 16 分連打で、実際プレイしてみるとエナジーゲージを全部削られるほどの連打ではないことがわかります。

かといって油断していると速攻で MISS × TAKE を出してしまう楽曲なので注意が必要です。

結んで開いて羅刹と骸 (BPM 127)

この曲は間奏の難易度は低く、通常の部分が難化しています。

EASY	★★★
NORMAL	★★★★★
HARD	★★★★★★★
EXTREME	★★★★★★★★★

たとえば、「らい らい 羅刹(らせつ)と骸(むくろ)」の部分の譜面が「→ ← ↓ ↑ ↑ ↓ ←」となっているのに対して、「↑」のターゲットマーカーがかぶって表示されているので難解な譜面に見えます。

また、各所に下図のような「△△△□□□△」のような 16 分連打が存在します。こちらの処理の方が厄介なのですが、エナジーゲージに大きな影響がないのでクリアだけを目指すのであれば無視してしまっても問題無いです。



Project DIVA 最新作攻略

パラジクロロベンゼン (BPM 132)

下表をご覧の通り★ 9 曲です。公式にも難曲とアナウンスされているのですが、この譜面は連打譜面ではなく比較的スローな譜面と読みにくい譜面で構成されています。

EASY	★★
NORMAL	★★★★
HARD	★★★★★★
EXTREME	★★★★★★★★★

その一例が下図ですが、「→× ↓ □ ←△ ↑ ○ → × ↓ □ ←←←」という譜面になっています。これを素直にやると指が疲れるので、すべて同時押しで処理すると良いでしょう。また、短めの長押しが頻繁に登場します。これについては、★ 8 の「こっち向いて Baby」にも短めの長押しが登場するので練習には良いのではないでしょうか。



裏表ラバーズ (BPM 159)

EASY	★
NORMAL	★★★★
HARD	★★★★★★
EXTREME	★★★★★★★★★

★ 9 曲の 2 つめです。下図のように「□ □□ □□ □ □□」のように終始不規則な連打が続けます。譜面上特に急激に変化する場所等は存在しないため、連打が得意な人はすぐにクリア出来るのではないかでしょうか。ちなみに私は連打が苦手なので今のところ完走できても CHEAP 判定です。



ローリングガール (BPM 195)

EASY	★★
NORMAL	★★★★★
HARD	★★★★★★★
EXTREME	★★★★★★★★★

全体的にはちょっと難化されたハイテンポな★ 8 曲ですが、部分部分で連打を要求される譜面です。

序盤は割と余裕にクリア出来ますが、後半に行くにしたがって難解な譜面へと変化していきます。この難解な譜面について行けないと、完走しても CHEAP 扱いになってしまいます。

積乱雲グラフィティ (BPM 138)

EASY	★
NORMAL	★★★★
HARD	★★★★★
EXTREME	★★★★★★★

間奏連打曲です。△の 16 分連打がちょうど曲の真ん中辺りで発生します。

△と↑キーを連打しておけばソングゲージも全部削られると言うこともなく、それ以外に（EXTREME 譜面的に）難しい所は一切ないので安心してプレイしてください。



Project DIVA 最新作攻略

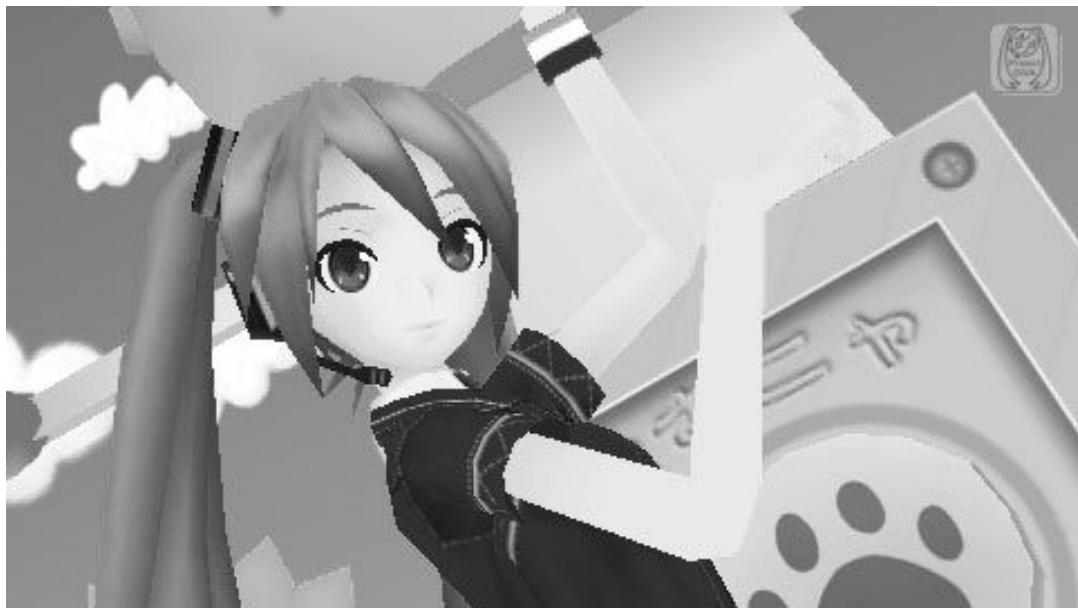
おわりに

かなり駆け足で Project DIVA -extend-についてご紹介してきましたがいかがでしたでしょうか？

この記事だけを見ると何のこっちゃわからぬ方もいらっしゃると思いますが、もし興味があれば PlayStation Store で体験版が公開されているのでプレイしてみることをおすすめいたします^{*13}。

もし、「〇〇の楽曲がわからない」等があれば読者アンケートに書いていただければ筆者のクリア出来る範囲で対応させていただきます。「クリア出来る範囲」と言いますのは、DIVA extend になってから難易度が上がっていて、筆者も EXTREME はあと 5 曲残ってる（クリア出来ていない）という由由しき事態が発生しています。~~とはいえ、なんか連打だけで難易度上げてるよう見え~~
~~るんですね~~

なお、この記事を書くに当たって、モジュール購入に必要なポイント等が掲載されている「初音ミク -Project DIVA- wiki^{*14}」を参考にさせていただきました。記事に載っていないような細かい情報も掲載されているので、困ったときに参考にすると良いと思われます。また、本記事に掲載したすべての画像の著作権は SEGA 様と Crypton Future Media, Inc. 様に帰属します。



本文にてミクさんの画像が足りなかつたことをここにお詫びいたします

*13 何らかの細工を PSP に施している方は体験版をプレイするにはファーム更新が必要になるので注意が必要です。もっとも、そのような人は自力で何とか出来ると思いますが

*14 <http://www19.atwiki.jp/mikudiva/>

GRな日々。 IX - 図書館探訪

文 編集部 葡萄酒

近頃、諸事情で中央図書館に入り浸る事が多かったのだが、中々興味深い部分を見つけたので紹介しよう。

筑波大学中央図書館は、ご存知の通り第一エリアに隣接している大きな建物である。筑波大学が開学した5年後の1978年に開館され、増築や図書館情報大学との統合を経て今に至る。蔵書数は和洋合わせて250万冊¹を超える、学内のみならず学外の利用者にも広く開放されている。サービスは図書の貸し出しのみに留まらず、セミナー室の利用やリファレンスデスク、文献の複写など多岐に及ぶ。

さて、図書館の内部であるが、一般的な利用者の大半は主にカウンターのある2階より上を利用している事と思う。我々が良く利用するであろう情報科学の資料は5階に収められているし、新聞や新書の類が配置してあるのは2階である。もちろん1階にも書架は設置されているのだが、どのような蔵書が収められているのかはご存知だろうか。



1階デスク



1回書架

筑波大学はその設立に際して東京教育大学を前身としている。東京教育大学もまた前身となった学校があり、その頃から受け継がれてきた資料はかなりの数になる。そのような古くからの資料が収められているのが1階、そして併設されている中2階なのだ。

中2階への階段



カウンター横の階段を下りると、少し開けた空間に出る。ここには新聞のバックナンバーの他、検索に利用する端末やコピー機、資料閲覧のためのデスクが用意されている。更に正面にある扉をくぐると、そこには貴重な資料が収められた書架たちが広がっている。

ここは移動書架になっており、他の階に比べると相当な密度で資料が収められている。筑波大学の図書館として集められた資料とは毛色が違い、外国の小説など文芸書も揃えられているのは嬉しい。

*1参考にしたのが21年度のデータであるため、23年度現在は更に増えているかもしれない。

GR Days IX

しい所だ。上記の通り、この上には中2階が併設されているので天井は低く、少し圧迫感がある。



中2階書架

では中2階に上がってみよう。1階の中程に、書架に埋もれるようにして階段がある。ただでさえ1階の利用者は少ないのだが、中2階については輪をかけて人気が無く、しんと静まり返っている。ここに収蔵されている資料は全体的に1階のものよりも古く、背表紙からタイトルが読み取れないもの、和綴じで背表紙が無いもの、損傷が激しく本としての体裁を保てていないものなど、時代を感じさせるものが多い。書架の間に立って古い紙の匂いに触れると、これだけ大量の情報が百年を超える昔から脈々と受け継がれてきたということが実感でき、圧倒される。

ざっと並んでいる資料を眺めると、大正14年の時刻表^{*2}や大戦中に書かれた戦況に関する報告などがあって興味深い。このような貴重な資料であっても貸し出し可能なものが多いので、興味のある方は実際に手に取って見るといいだろう。



中2階から1階を見下ろす



「汽車時間表 大正十四年七月号」

また、中2階の一部は吹き抜けになっており、書架を上から一望する事が出来る。こちらも中々壯観な眺めなので、一度足を運んでみる事をお勧めする。

さて、ここまで色々と紹介してきたが、もちろん1階以外にも面白い資料はたくさんある。館内にも本が読めるスペースは何カ所も用意されており、読書に没頭するには又とない好環境である。せっかく身近にこんな素晴らしい施設があるので、恩恵に与らない手は無いだろう。今まで図書館を利用した事の無かった読者の方は、この記事を機に図書館を利用してみるのも良いのではないだろうか。

*2 広域の路線図を見ると満州や樺太の路線が載っていたり、今とは違う漢字で表記されている駅名などがあつて面白い。

カーネルコンパイル虎の巻

文 編集部 zer0day

序文

こんにちは。zer0dayです。

今回は、Linux の中核である「カーネル」を改造するおはなしです。私の実験では、最小限の構成に絞ったカーネルをインストールすることによって、Debian の起動が 7 秒台になったり、消費電力が少なくなるなどの改善が見られました。

この記事ではカーネルコンパイルの基本的なことに加えて、distcc の使い方にも触れます。

さて、今回は私の手持ちの実験環境として、Debian 6.0 と ThinkPad X60 を使用しました。以下では Debian 6.0 での手順を紹介します。

準備しよう

まず、一般的な方法でインストールした Debian 6.0 を起動します。

準備作業は一部 root 権限で行いますが、コンパイル自体は root 権限が必要ないので、自分を src グループに追加して、/usr/src/ ディレクトリ以下を操作できるようにしておきます。

```
$ sudo usermod -a -G src `whoami`
```

※ここから root 権限を要するコマンドと通常ユーザのコマンドが混ざるので、シェルの頭 (#, \$) で区別します。

まず、カーネルのソースコードを入手します。

```
# apt-get install linux-source-2.6.32
```

次に、カーネルをビルドする上で必要なパッケージをインストールします。

```
# apt-get install fakeroot bzip2 kernel-package libncurses-dev  
# apt-get build-dep linux-source-2.6.32
```

入手したばかりのカーネルソースコードの在処へ移動し、ソースコードを開けます。

```
# cd /usr/src  
# tar xvjf linux-source-2.6.32.tar.bz2
```

グループ “src” が /usr/src に読み書きできるようにファイルパーミッションを設定します。

```
# chgrp -R src /usr/src
```

カーネルコンパイル虎の巻

```
# chown -R 775 /usr/src
```

展開したソースコードのディレクトリを、利便性のために /usr/src/linux というパスでアクセスできるようにします。

```
# ln -s linux-source-2.6.32 linux
```

※ root 権限での作業はここまで。

以降は、先ほど src グループに追加した通常ユーザで行います。

ビルドオプションを設定しよう

まず、以下のコマンドで作業ディレクトリをソースファイルの在処に移し、オブジェクトファイルや使わない config ファイルなどといったゴミを取り除きます。

```
$ cd /usr/src/linux  
$ make clean  
$ make mrproper
```

現在のカーネル設定を引き継ぎたい場合:

```
$ cp /boot/config-`uname -r` .config
```

完全にオリジナルの設定をしたい場合、 .config ファイルは形だけのものを作つてから次に進みます。

```
$ touch .config
```

いよいよ、ビルドオプションの選択画面に入ります。

```
$ make menuconfig
```

ここで、カーネルのビルドオプション選択インターフェースが出てきます。

最初はとんでもない量のオプションに圧倒されるかとは思いますが、 Gentoo Wiki^{*1} などを参考に設定項目をひとつおり見ていくとそのうち慣れます。

今回の話題とは少々ずれますが、カーネルを自前でコンパイルして使う Linux のうち著名なものに Gentoo Linux があり、Gentoo Wiki には各種ハードウェアに対応したカーネルビルドオプションの設定例が掲載されており、参考になります。この記事には個々のハードウェアに合わせた設定例は掲載しませんので、お試しになる場合は Gentoo Wiki 等でインストール対象のハードウェアの記事を探し、そのとおりに設定してみましょう。

*1 http://en.gentoo-wiki.com/wiki/Main_Page

さて、ひととおり設定ができたら、設定メニューを離脱し、新しい設定情報を `.config` に保存します（保存ダイアログが出るまで `Exit` を選択し、ダイアログで `Enter` を押せば大丈夫です）

distcc を導入しよう

いよいよコンパイルですが、コンパイルの高速化のために `distcc` を導入してみましょう。

`distcc` とは、ネットワーク上のマシンにコンパイルを分担させることで、コンパイルの速度向上を図るソフトです。例えば私の環境では、ThinkPad X60 単体では 1 時間ほどかかったコンパイルが、手持ちの Core i5 マシンを `distcc` に参加させて 2 台がかりでコンパイルしたところ、20 分程度にまで短縮できました。なお、`distcc` の導入は必要ではありませんので、興味があるか、またはコンパイルを実行しようとしているマシンが非力なためコンパイルに数時間から数日かかりそうだという場合に試してもらうだけで結構です。

必要なものは次のとおりです：

- `distcc`
- ネットワーク経由で接続できる `distcc` サーバ（私の実験では Ubuntu 10.04.3 を使いました）
- クライアントとサーバ両方に、バージョンの一致する `gcc`（クライアントに合わせます）

まず、`distcc` クライアントをインストールします。

```
# apt-get install distcc
```

次に、`distcc` を使うために、「マスカレードディレクトリ」を作成します。

```
# mkdir -p /usr/local/distcc/bin  
# cd /usr/local/distcc/bin  
# ln -s /usr/bin/distcc c++  
# ln -s /usr/bin/distcc cc  
# ln -s /usr/bin/distcc g++  
# ln -s /usr/bin/distcc gcc
```

作成したマスカレードディレクトリを反映させ、ついでに各種オプションを定義します。

```
$ export CONCURRENCY_LEVEL=10  
$ export DISTCC_HOSTS="10.0.1.6 localhost"  
$ export PATH="/usr/local/distcc/bin:${PATH}"
```

ここで、例として `CONCURRENCY_LEVEL` を 10 としましたが、この `CONCURRENCY_LEVEL` とは、`make` の `-j` オプションと同じものです。私の実験環境下では、`localhost` が Core 2 Duo T5500、すなわち最適な `-j` オプションは 2 であり、もう一方で `distcc` サーバ（10.0.1.6）がハイパースレッディング有効の Core i5 650 なので最適な `-j` オプションは 8 なので、合わせて 10 としました。また、`DISTCC_HOSTS` には、スペース区切りで優先度の高い順に並べたホスト名または IP アドレスを指定します。上の例では、リモートの `distcc` サーバは 10.0.1.6 となっていますが、ここは各自のネットワーク構成によって異なります。また、`distcc` サーバ側でどれほどのジョブス

カーネルコンパイル虎の巻

レッドを受け取るかも後述するファイルから設定できるので

最後に、

```
$ gcc --version
```

とし、`gcc` のバージョンを控えておきます。おそらく `gcc 4.4.5` が搭載されているでしょう。

これでクライアント側の設定は完了です。さらに、サーバ側の設定をします。

まず、サーバにログインします。

以下のコマンドはサーバ側で実行します。

```
# apt-get install distcc

# nano /etc/default/distcc
--編集する行--
STARTDISTCC="false"           ← true に変更
ALLOWEDNETS="10.0.1.0/24"     ← 接続を許可するネットワークを指定
JOBS=""                       ← 受け取るジョブスレッド数。空なら無制限
--編集ここまで--
```

クライアント側と `gcc` のバージョンが一致しているか確認します。

```
$ gcc --version
```

一致していたら、`distcc` を開始できます。

```
# service distcc start
```

これで、`distcc` の準備が完全に整いました。

コンパイルしよう

いよいよコンパイルを実行します。

クライアントに戻り、次のコマンドを実行します。

```
$ make-kpkg clean
$ make-kpkg fakeroot make-kpkg --append-to-version "<-suffix>"
--revision "<revision#>" --us --uc --initrd kernel_image kernel_headers
```

この `suffix` と `revision#` は自分で決めてください。私は ThinkPad 用のカーネルを初めて作ったので、`suffix` を `thinkpad` とし、`revision#` を 1 としました。この場合のコマンドは、

```
$ make-kpkg fakeroot make-kpkg --append-to-version "-thinkpad"
--revision "1" --us --uc --initrd kernel_image kernel_headers
```

となります。

インストールしよう

コンパイルが無事完了すると、1つ下のディレクトリ、すなわち /usr/src に 2つの .deb パッケージが生成されているはずです。その2つのパッケージを dpkg を使って通常の .deb のようにインストールし、update-initramfs コマンドを使って新しいカーネルイメージを反映させます。

```
$ cd /usr/src  
# dpkg -i linux-image-2.6.32-thinkpad_1_amd64.deb  
# dpkg -i linux-headers-2.6.32-thinkpad_1_amd64.deb  
# update-initramfs -c -k 2.6.32-thinkpad
```

最後に、新しくカーネルをインストールしたマシンを再起動し、GRUB から新しいカーネルを選択します。

おわりに

うまくいきましたか？
これでカスタムカーネルの導入は完了です。よい魔改造ライフを！

実用 Perl6

文 編集部 葡萄酒

はじめに

前回までは Perl6 の愉快な機能を紹介してきましたが、今回は Perl6 を使って実用的(?)なプログラムを書いて行きましょう。具体的には、以下の仕様を満たすプログラムを目指します。対象とする環境は執筆次点での最新版の Rakudo (2011.11-60-g2154ebb) です。

- * 以下の関数を渡して、そのグラフをビットマップで出力する
- * 実数を 1 つ渡して、実数が返ってくる関数(例: $y = \sin(x)$)
- * 実数を 2 つ渡して、真理値が返ってくる関数(例: $x + y < 1$)
- * 関数に定義域を指定できるようにする

また、サンプルコードの簡略化のため、原点は中央で固定します。

Bitmap クラス

余り長い記事になっても疲れてくると思いますので、ビットマップファイルへの出力部分にはあまり触れない方針で行こうと思います。ソースコードは公開していますので、詳しい実装を見たい方は GitHub¹ を参照してください。この Bitmap クラスを継承して Graph クラスを書いていきます。

まず前提として Bitmap クラスの提供するメソッドは以下の 5 つです。

```
method new (Int $width where { $_ > 0 }, Int $height where { $_ != 0 })
# コンストラクタ。引数には出力する画像のピクセル数を指定

method write (Str $file)
# 引数のファイル名への書き出し

method getpixel (Int $x where { 0 <= $_ < $!header.width },
                Int $y where { 0 <= $_ < $!header.height.abs })
# 指定した座標の色情報を取得

method setpixel (Int $x where { 0 <= $_ < $!header.width },
                 Int $y where { 0 <= $_ < $!header.height.abs },
                 Int $b where { $_ ~~ 0..255 },
                 Int $g where { $_ ~~ 0..255 },
                 Int $r where { $_ ~~ 0..255 })
# 指定した座標の色情報を設定
```

¹ <https://github.com/VienosNotes/p6-Bitmap>

```

method fill (Int $b where { $_ ~~ 0..255 },
             Int $g where { $_ ~~ 0..255 },
             Int $r where { $_ ~~ 0..255 })
# 画像全体を指定した色で塗りつぶす

```

また、画像の情報を保持する属性として `Header` クラスのインスタンスを保持しています。中身は色々はありますが、`Bitmap.new` に渡した幅と高さが格納されていると考えて差し支えありません。

Graph クラス

では、`Graph` クラスの設計に入っていきましょう。上記の仕様を満たすために必要と思われるメソッドと属性は以下の通りです。

```

has Code @.funcs
# 設定された関数を保持

has $.x_limit is rw = 1
has $.y_limit is rw = 1
# 画像端の座標を保持
# デフォルト値は 1

method set_func
# 描画する関数を設定

method draw
# 設定された関数を計算して、グラフを描画
# axis オプションが指定されたときは座標軸も描画

```

set_func メソッド

`Graph` オブジェクトに描画する関数を設定するメソッドです。

このプログラムは二次元のグラフしか対象としないので、設定する関数が受け取る引数の数を 1 つ、または 2 つに制約します。上の例に既に出てきていますが、引数に値の制約を掛けるには、シグネチャに `where` 節を指定します。この場合だと、以下のようになるでしょう。

```
method set_func (Code $f where { $_.arity == 1|2 }) { ... }
```

関数オブジェクトから引数の個数を取得するには `arity` メソッドを使います。`where` 節の中では制約の対象となる引数は特殊変数 `$_` に格納されているので、`$_.arity` が 1 または 2 であるかどうかをチェックしています。また、Perl6 に慣れていない読者の方は `1|2` という表記に多少違和感をおぼえるかもしれません、Perl6 では `Junction` という機能を使って「1 または 2 である」という状態を値として保持できるようになっています。`Junction` は普通の関数への引数として渡す事も可能で、その場合は `Junction` を受け取った関数が自動的に並列に実行される

Pragmatic Perl6

(**Junctive Autothreading**)など面白い仕組みになっているのですが、やはり長くなるので割愛します。

ではメソッドの中身です。これは受け取った引数を`@.funcs` に追加するだけなので、とても簡単ですね。以上を纏めると、`set_func` メソッドの実装は以下のようになります。

```
method set_func (Code $f where { $_.arity == 1|2 }) {
    @!funcs.push($f);
}
```

draw メソッド

今までには前置きのようなもので、ここからが本番です。少し唐突ですが、この `draw` 関数に座標軸を描画するオプションを追加しようと思います。関数のオプションは、省略可能 (optional) な名前付き引数を使って以下のように定義します。

```
method draw (:$axis?)
```

このようにシグネチャの仮引数の先頭にコロンを付ける事で、名前付き引数を定義することができます。また、末尾にクエスチョンマークを付ける事で、省略可能な引数を定義できます。これらを組み合わせる事で、以下のようなメソッド呼び出しが可能になります。

```
$graph.draw(:axis)
```

これは前の記事でも紹介した通り、`axis` という名前付き引数に 1 を渡す糖衣構文です。つまり、メソッド側で`$axis` の値をチェックする事でオプションが指定されたかどうかを判定できるわけですね。このメソッドでは、`axis` オプションが指定されていたときは、グラフを書く前に`draw_axis` メソッドを内部的に呼び出すようにします。

このメソッドの処理の流れとしては、「全体を白で塗りつぶす」「オプションを見て座標軸を描画する」「グラフを描画する」となりそうです。実際に関数を計算するのは別のメソッドに分けるとすると、`draw` メソッドの実装は以下のようになります。

```
method draw (:$axis?) {
    self.fill(255,255,255);
    if $axis {
        self.draw_axis;
    }
    for @!funcs {
        self.calc($_);
    }
}
```

draw_axis メソッド

一番重要な calc メソッドは取りあえず置いておいて、先に draw_axis メソッドから実装して行きます。このメソッドは引数も取りませんし、画像の真ん中に十字を引くだけなので簡単ですね。実装は以下の通り。

```
method draw_axis {
    my $center_x = (!$header.width / 2).Int;
    my $center_y = (!$header.height / 2).Int;

    for ^$!header.height {
        self.setpixel($center_x, $_, 128, 128, 128);
        # グラフと被ると見辛いので灰色
    }

    for ^$!header.width {
        self.setpixel($_, $center_y, 128, 128, 128);
    }
}
```

for 文の対象として`^$var` という記法が使われていますが、これは0から\$var-1までの Range を生成する糖衣構文です。これを用いる事で、特定の回数だけループを回すような処理が簡潔に記述できます。

ここで簡単に for 文の紹介をしておきましょう。Perl6 の for 文は C 言語のそれとは違い、リストに対してイテレートを行うための構文です。基本的な使い方は Perl5 での for/foreach と同じですが、C 言語スタイル(3-part for)のループは記述できません²。for に続けてイテレートしたいリストを記述すると、それぞれの要素が`$_`にコピーされてブロック内の処理が行われます。`$_`ではなく明示的に名前を付けたい場合は、以下のように pointy block という構文を使うことで実現できます。

```
for @list -> $element { ... }
```

また、ブロックの中でリストの要素を直接変更したい場合は、`rw` トレイトを指定する事でコピーではなく参照が設定されます。

```
for @list -> $element is rw { ... }
```

²かわりに C 言語スタイルのループを書くには loop 文を使います。

Pragmatic Perl6

calc メソッド

お待ちかねの calc メソッドです。これは引数として受け取る関数の種類で処理を切り替えたいので、そもそも別の関数に分けてしまいましょう。シグネチャは以下の通りです。

```
multi method calc (Code $f where { $_.arity == 1 }) { ... }
multi method calc (Code $f where { $_.arity == 2 }) { ... }
```

これで引数の個数によって実行時に関数を呼び分けられます(multiple dispatch)。まずは1引数の関数を受け取る方からです。文章で説明するのは難しいので、実装を先に見て頂きましょう。

```
multi method calc (Code $f where { $_.arity == 1 }) {
    my @val;
    my $width = $!header.width;
    my $height = $!header.height;

    for ^$width {
        my $x = -$!x_limit + (1 / ($width) * 2 * !$x_limit) * $_;
        # 画像上の座標をグラフ上の座標に変換

        @val = ($f($x.Num)).list;
        # 関数に渡す

        CATCH { next; }

        for @val -> $y {
            my $h = ($height / 2 + ($y * $height / (2 * !$y_limit))).Int;
            self.setpixel($_, $h, 0, 0) if $h < $height;
            # y 座標が画像の中に入っていれば setpixel に渡す
        }
    }
}
```

画像の左から順番に横位置を x 座標に変換して、受け取った関数に渡して計算していきます。返って来た y の値を受け取ったら、逆に画像上の座標に変換して、setpixel に渡して点を描画して行きます。右端まで処理が進んだら終了です。この関数のキモになるのは、内側の for ブロックにある CATCH 節です。これは所謂例外処理の一つで、ブロックの中で起きた例外を検出すると CATCH 節に処理が移ります。

これは、描画する関数に定義域を設定できるようにするためのものです。あらかじめ描画する関数のシグネチャに where 節を用いて x 座標の範囲を指定しておく事で、その範囲外での呼び出しは失敗して例外が発生し、CATCH 節によって次の座標へと処理が進むことになります。具体的には、以下のような形で関数が設定できるようになります。

```
$graph.set_func(sub ($x where { 0 <= $_ < 10 }) { return 2 * $x }
# 0 以上 10 未満の範囲で y=2x のグラフを描画
```

1引数の方はこれでいいでしょう。次は2引数の方です。お察しの通り、シグネチャは where 節の中身が変わっただけですね。

```
multi method calc (Code $f where { $_.arity == 2 }) { ... }
```

手抜きですが、受け取る関数は x,y 座標が渡された時に条件を満たすかどうかを判定するものとします。画像上の座標とグラフ上の座標がきちんと対応しないため、線のグラフは殆どの場合うまく描けませんが、「範囲に入るかどうか」という関数だと大体綺麗に描けます。

```
multi method calc (Code $f where { $_.arity == 2 }) {
    my $width = !$header.width;
    my $height = !$header.height;

    for ^$width -> $w {
        my $x = -$!x_limit + (1 / ($width) * 2 * !$!x_limit) * $w;

        for ^$height -> $h {
            my $y = -$!y_limit + (1 / ($height) * 2 * !$!y_limit) * $h;
            self.setpixel($h, $w, 0, 0, 0) if $f($x, $y);

            CATCH { next; }
        }
    }
}
```

方針は殆ど 1引数の場合と変わらないので、特筆すべき事もありません。

使ってみる

さて、必要な機能は一通り実装し終わったので、纏めてみます。

```
use v6;
use Bitmap;

class Graph is Bitmap {

    has Code @.funcs;
    has Header $.header;
    has $.x_limit is rw = 1;
    has $.y_limit is rw = 1;
```

```
method set_func (Code $f where { $_.arity == 1|2 }) {
    @!funcs.push($f);
}

method draw (:$axis?) {
    self.fill(255,255,255);

    if $axis {
        self.draw_axis;
    }

    for @!funcs {
        self.calc($_);
    }
}

method draw_axis {
    my $center_x = ($!header.width / 2).Int;
    my $center_y = ($!header.height / 2).Int;

    for ^$!header.height {
        self.setpixel($center_x, $_, 128, 128, 128);
    }

    for ^$!header.width {
        self.setpixel($_, $center_y, 128, 128, 128);
    }
}

multi method calc (Code $f where { $_.arity == 1}) {
    my @val;
    my $width = $!header.width;
    my $height = $!header.height;

    for ^$width {
        my $x = -$!x_limit + (1 / ($width)*2*$!x_limit) * $_;
        @val = ($f($x.Num)).list;

        CATCH { next; }

        for @val -> $y {
            my $h = ($height/2 + ($y * $height / (2*$!y_limit))).Int;

```

```

        self.setpixel($_, $h, 0, 0, 0) if $h < $height;
    }
}
}

multi method calc (Code $f where { $_.arity == 2 }) {
    my $width = $!header.width;
    my $height = $!header.height;

    for ^$width -> $w {
        my $x = -$!x_limit + (1 / ($width) * 2 * $!x_limit) * $w;

        for ^$height -> $h {
            my $y = -$!y_limit + (1 / ($height) * 2 * $!y_limit) * $h;
            self.setpixel($h, $w, 0, 0, 0) if $f($x, $y);

            CATCH { next; }
        }
    }
}
}

```

では実際に使ってみましょう。`set_func` に渡す関数の作り方は色々あるのですが、ここでは `whatever code` と `pointy block`³ を使うようにしています。描画範囲は x,y ともに-2 から 2 までです。

```

my $g = Graph.new(200,200);

$g.set_func: *.sin;
$g.set_func: do -> $x where { $_.abs < 1 } { $x ** 2 };

$g.x_limit = 2;
$g.y_limit = 2;

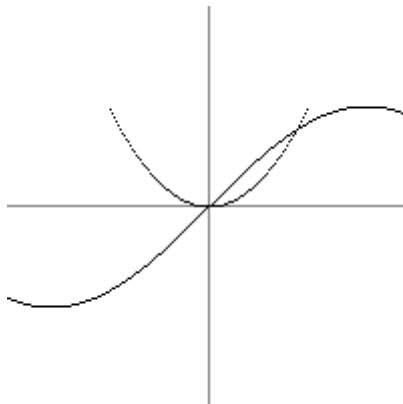
$g.draw(:axis).Str;
$g.write("graph1.bmp");

```

*3 詳しくは Synopsis (<http://perlcabal.org/syn>) を参照。

Pragmatic Perl6

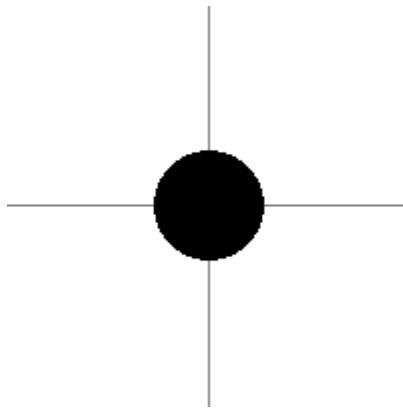
出力結果: $y = \sin(x)$ のグラフと、 $(-1 < x < 1)$ の範囲で $y = x^2$ のグラフが描画されています。



2引数を受け取る方も描画してみます。

```
$g.set_func: do -> $x, $y { $x ** 2 + $y ** 2 <= 0.3 };
```

出力結果: 半径ルート 0.3 の円が描画されています。



正しく出力できているようですね。

まとめ

ここまで駆け足でしたが、お楽しみいただけたでしょうか。処理系の実装もかなり進んできたので、色々なコードが正しく動くようになってきましたし、速度自体も実用レベルに近づいてきています。まだまだ面白い Perl6 の機能はありますので、読者の皆さんにも是非 Perl6 のパワフルさに触れてもらいたいと思っています。

ここで宣伝のような形になるのですが、特定のテーマについて 12 月 1 日から 25 日まで日替わりでブログを描いていく Advent Calendar という企画があるのはご存知でしょうか。今年は私が主催で Perl6 Advent Calendar を開催しているのですが、参加者が圧倒的に足りません。記事を書いてくれる人は絶賛募集中ですので、興味のある方は是非以下の URL にて登録をお願いします。

<http://atnd.org/events/22820>

はじめに

前回は量指定子とゼロ幅マッチについて話しました。このテーマについて書いてくれ、とかいうものはなかったので、予告どおりバックトラックと最適化についての話をします。また前回は Vim の表記を書きましたが、今回は主に締切の関係で Perl 表記のみを記し、Vim 表記は割愛させていただきます。本当にすみません^{*1}。

バックトラックとは

バックトラックは古くから存在する概念です。いくつかの例を挙げて概要を紹介したいと思います。

ヘンゼルとグレーテルという話があります。知らない方はまずいないと思いますが、一応ここで重要な部分の内容を述べます。

貧しい家にヘンゼルとグレーテルという二人の子どもがおりましたが、口減らしにため、彼らは両親に薪拾いと称して森の深くへと誘いこまれ、遭棄されてしまいました。しかし、予めその計画を知っていた兄のヘンゼルは、森に入る道中、目印となる道に小石を置きました。それによって、深い森の中から目的の家まで無事に帰りつくことが出来ました。

ここでヘンゼルが行なった行動というのは、道の分岐が発生した時に、目印を付けるというものです。この様にすることで、ある道へ進んで、その後何かの理由で引き返す必要が生じた時、円滑に目的の分岐まで引き返すことが可能となります。

また現代においても、ギャルゲなどの読むゲームの中で、選択肢の直前でクイックセーブすると聞きます。選択肢を分岐と捉えれば、分岐に目印を付けフラグ回収などの目的に応じて引き返すことが可能となるわけですので、バックトラックを行っていると考えられます。

難しい言葉で言えば、バックトラックは深さ優先探索の一つです。

ヘンゼルとグレーテルでは小石、ギャルゲではセーブデータが該当する「目印」ですが、正規表現の世界ではこれを選択ポイント^{*2}と呼びます。選択ポイントは基本的にスタックであるので、マジック・ザ・ギャザリングというカードゲームをやったことのある人は分かると思いますが、後に積まれた方が先に出されるという構造を取っています。

*1 Vim には正規表現のデバッガがないというのも大きいです。

*2 選択ポイント (saved choice point) という用語は、Perl6 の仕様で記述されている単語です。この記事の参考書である「詳説 正規表現」やその原著、「Mastering Regular Expressions」をあたたかく、保存ステート (saved state) という単語が用いられていました。saved choice point、saved state を検索してみたところ、前者は JavaCC、後者は.NET に関する MSDN のドキュメントで使われていると分かりました。しかし、それ以外の公式ドキュメントなどにおいて、saved choice point や saved state という用語が使われている形跡はありません。なのでこの単語はマニアックな人を除いて、一般的ではないと考えて使うべきかと思います。この記事では後々を考慮し、Perl6 の仕様で採用されている選択ポイントを用いますが、意味はどちらも同じです。

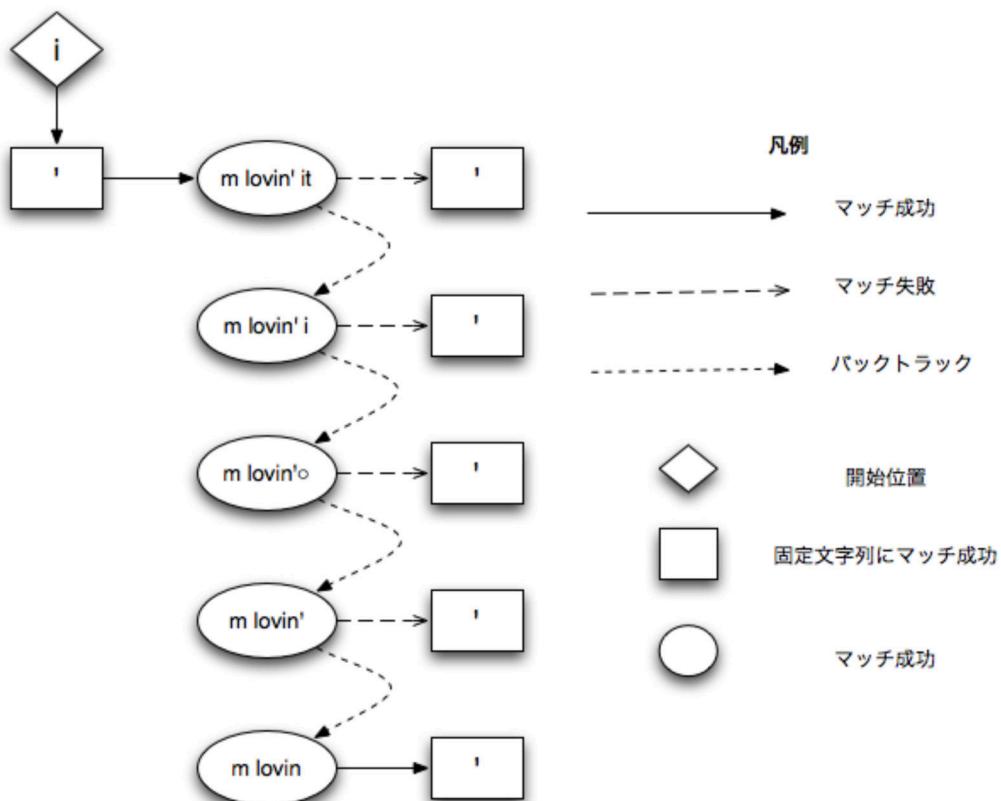
regex I

バックトラックと正規表現

正規表現の中にも分岐が存在し、それは以前紹介した量指定子や選択です。例えば、「i'm lovin' it」に対して、「.*?」という正規表現を試す場合、正規表現のエンジンは先頭にある「i」で//のマッチに失敗し、次に「」にマッチしたあと、「.*?」というサブパターンを解釈します。

貪欲な「.*?」は「m lovin' it」にマッチして、それ以降の文字が無くなります。サブパターン「.*?」のマッチが終了したので、次に//をマッチしようと試みますが、文字が無く当然マッチしないので失敗します。

ここで正規表現エンジンはバックトラック出来るかどうかを判定します。「.*?」は「m lovin' it」にマッチする間、文字なしから「m」、「m ○³」、「m l」、「m lo」などの文字列全てにマッチ出来るので、その各々へ戻れるように選択ポイントを生産していました。ということは、現在選択ポイントは13個存在しています。従って、正規表現エンジンはバックトラックします。この時、バックトラックした選択ポイントは捨てられます。最も新しく蓄積された選択ポイントは「t」の前なので、「.*?」を「m lovin' i」にマッチさせ、//がその次の文字にマッチするか試みますが、「m lovin' i」の次は「t」なので、//にはマッチせず失敗します。という感じで次々とマッチするまで選択ポイントを遡ります。



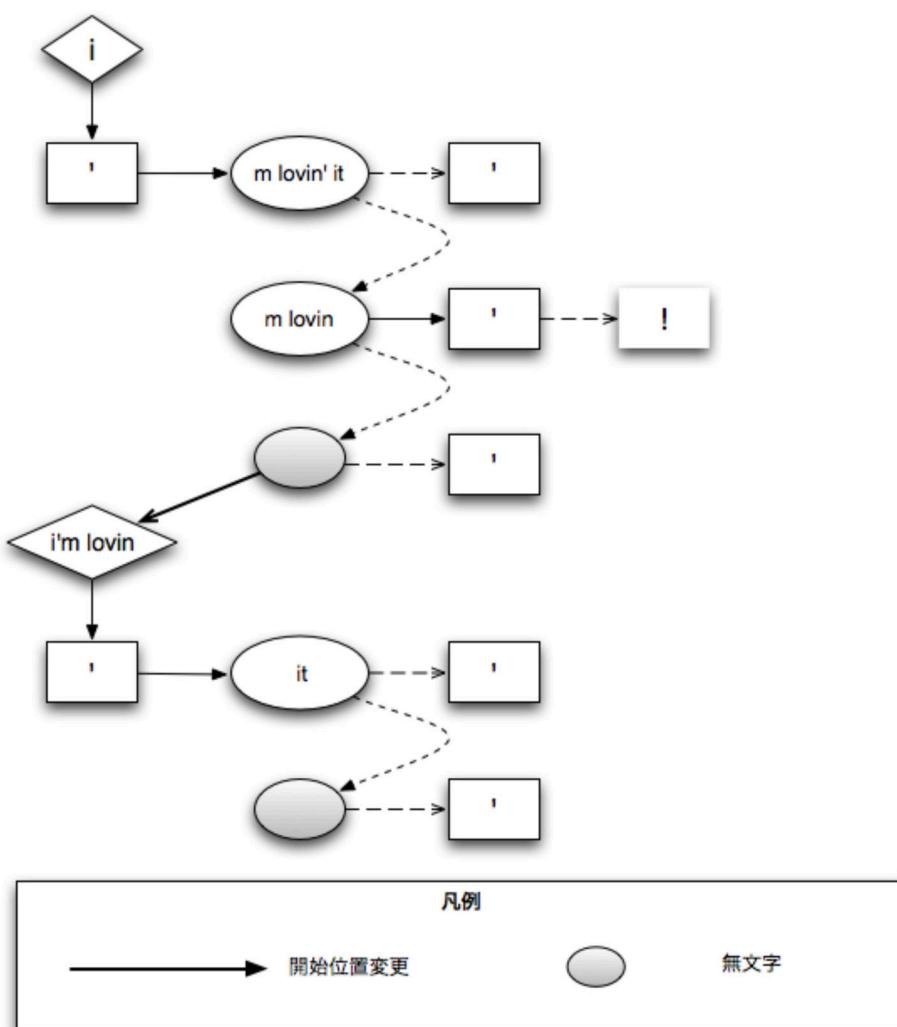
*3 このように、スペースの存在がわかりにくい場合は、適宜スペースを○に置き換えます。

こうして、マッチングの結果「'm lovin'」という文字列が手に入ります。このように次々と選択ポイントへ戻って試して行き、`/*`が「m lovin」にマッチした時に、`//`が「」にマッチして成功となります。

正規表現の失敗

次は先程の「i'm lovin' it」に対して`/.*/!`という正規表現を適用してみます。「i'm lovin' it」の中に「!」はないので、サブパターン`!/?`にマッチするものがありません。従ってこのマッチングは失敗するのですが、これもまた面倒なプロセスを経て失敗します^{*4}。

まず、先程のように図^{*5}を見てみます。



*4 少なくとも Perl の正規表現エンジンは、マッチングの前に最適化処理が行われるので、実際にマッチングすら行われない可能性がありますが、この記事では無視します。

*5 先程の図と比べて、量指定子が行なったバックトラックは一部割愛しています。

regex I

先程と同じように「i」は//にマッチしないので、正規表現のエンジンは「」からマッチを開始し、「」は//にマッチするので、マッチが継続されます。貪欲な/*が飲み込んだ部分を捨てながら、サブパターン/*!は「m lovin'」とマッチします。しかし、正規表現全体は/*!なので、直後に//とマッチしなければなりませんが、この後の文字はスペースなのでマッチに失敗します。

よって、/*!はさらに選択ポイントを遡り、文字なしまで行きますが、「」すらないのでまた失敗です。

選択ポイントを使いはたすと、次に正規表現エンジンはマッチングの始点を変更します。/*!がマッチするためには、「」の直前の位置から開始する必要があるので、正規表現エンジンはそこまで前進して、再度マッチを試みます。「」があるのは、「i'm lovin」の直後なので、そこからマッチングを開始し、「」が//にマッチし、残りの/*!がマッチするかどうかを試行します。この時/*!は、「it」、「i」、「○」、「文字無し」にマッチし、選択ポイントを積み上げます。しかし、どの選択ポイントであっても直後に//はマッチしないので失敗し、また選択ポイントが尽き果てます。

また選択ポイントが失われたので、正規表現エンジンはさらに開始位置を前進させますが、もはや//がマッチする位置すら見つからず、正規表現はここでマッチングの失敗を報告します。

このように、人間が見れば失敗が明らかなマッチングであっても、正規表現エンジンは選択ポイントが残っている状態でマッチングを止めるという動作を通常は行ないません⁶。

選択ポイントの山

正規表現のマッチングは遅い、という話をどこかで聞くかと思います。先程の失敗例を見れば、正規表現が遅いのも理解出来るかと思います。しかし、正規表現エンジンは頭の良い人達が寄つて集つてがんばった産物なので、先程の、「i'm lovin' it」に対して行った/*!のマッチ失敗なども一瞬で報告されます。

もっさり感を出すためにより意地悪なケースを使うことにしました。「=XX=====」に対して、/**X(+)+X**⁷を試行してみます。

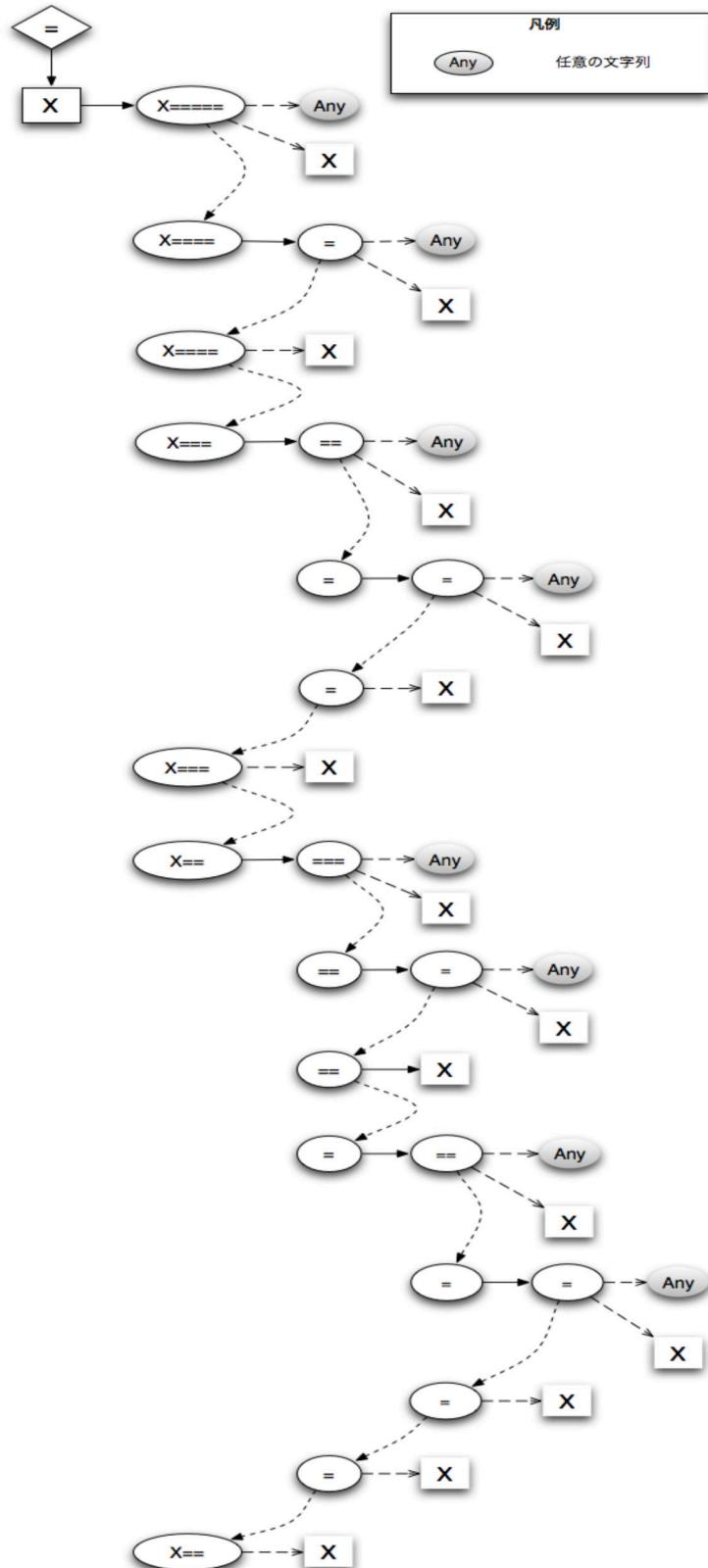
これについても図を用意したのですが、後述するように「=」の数が増えると図が巨大になり、到底収まらないので、「=XX====」という文字列に対する適用結果を図にしました。

*6 Perlの正規表現エンジンは、(*COMMIT)と(*ACCEPT)という特殊なバックトラック制御記号によって、任意の場所でマッチングを終了出来ます。というのも(*COMMIT)はそれまでに積んだ選択ポイントを全て破棄し、その後自身の場所へバックトラックして来た場合は、そこで失敗を報告します。(*ACCEPT)は(*ACCEPT)に到達した時に直ちにマッチングの成功を報告します。(*COMMIT)はPerl6の<commit>あるいはトリプルコロン(::)と若干動作が異なるので使う場合は注意してください。

*7 見にくいかかもしれません、これは連続したイコール(=)です。

*8 前号で話した通り、Perlの正規表現におけるグループ化は/(?: hoge hoge)/ですが、これも可読性を落とし、本質的ではないので/(hoge hoge)/と表記します。

*9 Perlの場合、/**X(+)+X**とすると正規表現の最適化によって、マッチングの前に失敗が報告されるので、検証では/[=:](?:+)+[=:]/という正規表現を用いました。しかしこれは本質的ではないうえに、可読性を損なうので/**X(+)+X**と表記します。



regex I

これは実際途中までしか書かれてません。しかしこれ以上図を書くのが死ぬほど面倒になったので止めました。とりあえず、この図から問題は明らかだと思います。バックトラックする場所が徐々に増えてゆくことです。これはサブパターン $/(.+)/$ に問題があり、 $/(.+)/$ とは、 $/(.)/$ にマッチしたものを一つ以上繰り返すわけではなく、単純に、 $/(.)/$ を一つ以上繰り返すわけなのでこの様になります。

例えばこの記事を書くために使っている Vim というエディタも、

「=XX=====」と書かれた行に対して、 $/X\$(\$\$\$)\$\$+X$ などと検索すると固まってしまいますし、Ruby 1.9.2p180 でも試したところ固まりました。

また、JavaScript もブラウザによってはこの影響を受けるようです。このような正規表現を含むページに Twitter で色々な人々にアクセスしてもらい、どのような結果になるのか調べました。結果的には IE、Chrome は大変な時間がかかるてしまうということでした。

ただちに処理出来るもの、出来ないものと色々ありますが、ここで注意したいのは、基本的に短時間で処理出来るものが特殊であるということです。先程の図の通り、正規表現の構造上^{*10}、このようなマッチングは失敗するまでに非常に時間がかかるはずです。

Perl や PHP、Firefox の JavaScript はどうしてこれほど高速なのかは謎ですが、恐らく何らかの最適化処理が施されている結果と思われる所以、Chrome などが固まるからバグというわけではありません。

やばい正規表現の具体例

エディタや一部のブラウザなどを苦しめる正規表現があるということは分かったと思いますが、一般的にイコールと「X」という文字列に対してマッチングするという場面は考えにくいです。そこで、もう少しやりがちな例を紹介しましょう。今、シングルクオート（「'」）で囲まれた文字列にマッチする正規表現を考えたいとします。つまり、「i'm lovin' it」に対して使えば、「m lovin」にマッチするというものです。これは序盤で挙げた $/\.\+/'$ で良さそうですので、さらに条件を付け加えて、「¥」でエスケープされたシングルクオートは無視するようにしましょう。つまり、「¥m lovin¥ it」であれば、「i¥m lovin¥ it」がマッチすれば良いわけです。

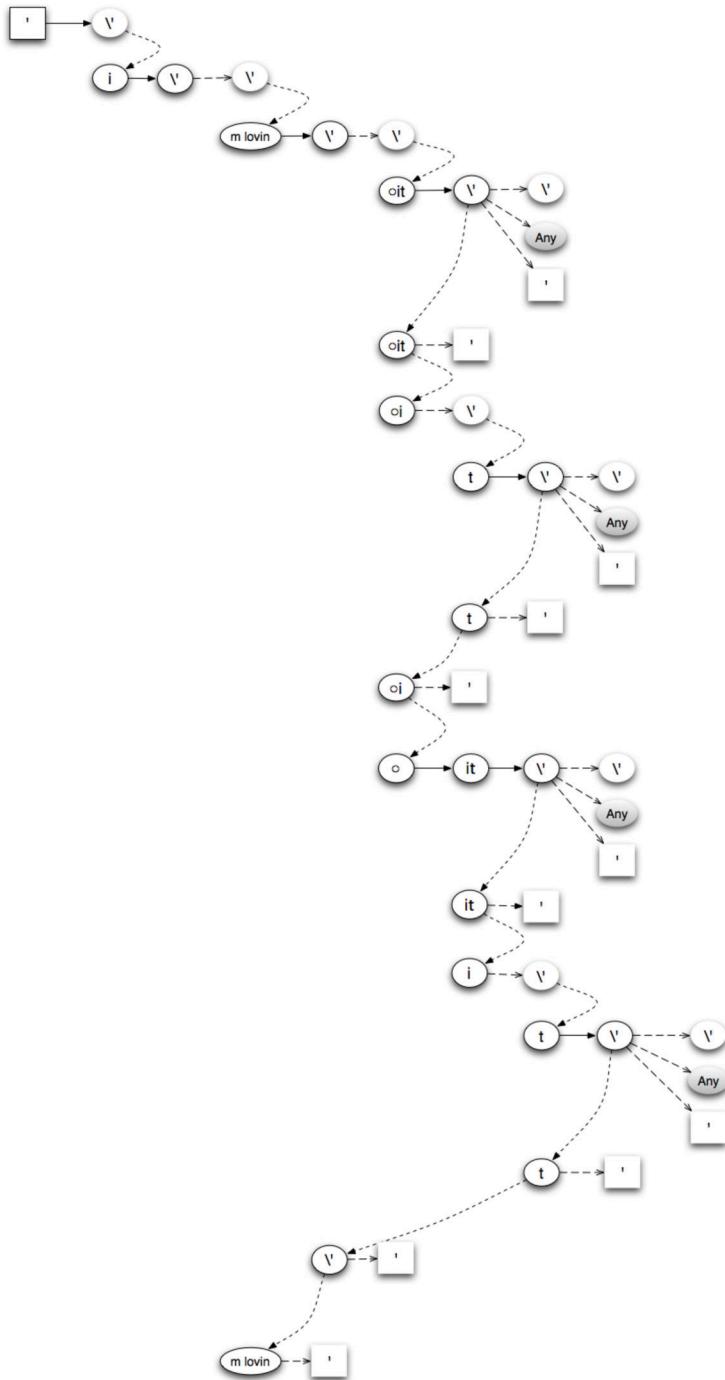
これはクオートの中に存在して良い文字を考えれば良いのです。一つはエスケープされた文字 $"$ 、つまり $\$\$$ 。そしてエスケープでもクオートでも無い文字列、つまり $[\^¥\$]$ です。この二つのどちらかであればクオート内に存在して良いので、選択を使えば良いということで $\$\$|[\^¥\$]'$ となります。これらで構成された文字列がクオートで挟まれていればよいので、最終的な正規表現は $/"(\$\$|[\^¥\$])+"/$ となります。

これは前章の $/(.+)/$ より酷いことになりそうです。なぜなら、 $/(A|.+)*/$ という構造になっているからです。選択が生み出す選択ポイントと、量指定子が生み出す選択ポイントが重なって、とんでもない量の選択ポイントを積み上げます。

*10 正確には NFA という正規表現エンジンの特徴です。NFA については Wikipedia などを参考にしてください。

*11 どうしてエスケープされたクオートではないかというと、エスケープされたクオート以外の文字列を現すのは大変だからです。

試しに「'm lovin' it」に`"(\w|[^"]")+"*`を適用した図^{*12}を見てみましょう。「'm lovin' it」は最後のクオートすらもエスケープされているので、閉じるクオートが無く失敗します。



*12 この図で用いられている Any は、「エスケープされた文字でも、クオートでもない文字列」です。また図中では「¥」の代わりにバックスラッシュが使われています。

regex I

前頁の図は中途半端ですが、図がデカくなりすぎるのでこの程度とします。見てのとおりたかだかこの程度にも関わらず、この後「m lovin」の部分を次々と分割してゆき、とんでもない処理量となります。

ここまでで、なんとなくやばい要素が浮き彫りになってきたかと思うので、良くない要素を次にまとめてみます。

- 選択を量指定してしまう。
- 量指定子を量指定してしまう。

なのでこれらをなんとかする方法を紹介しようと思います。

選択の適用順序

*/ (kadofuji|kadofuji makudonarudo) /*などという正規表現を、「kadofuji makudonarudo」に適用した場合、Perl など多くの処理系に実装されている正規表現エンジン^{*13}においては、選択の左側を優先するので、「kadofuji makudonarudo」の「kadofuji」の部分にマッチします。選択は一番左にあるサブパターンに失敗すると、一つずつ右のサブパターンを試してゆきます。

つまり、適用する文字列にだいたいの想像がつく場合は、出現頻度のより高いものをより左側へ置いた方が速くなる可能性があります。

クオートで囲まれた文字列の例でも、一番多いのは恐らくエスケープでもクオートでも無い文字列ですので、クオートされた文字のサブパターン/`\$``/を左に置いた/`\$`|[^$`]+`/よりも、普通の文字列のサブパターン/`[^$`]+``/を左に置いた/`[^$`]+`\$``/の方が高速になると期待出来ます。

しかしこれも結局は気休めで、失敗するときは全てのパターンを探索してしまうので、選択の順番を入れ替えることで高速化するのは成功するときだけとなります。

選択ポイントと量指定子

前号でも紹介した量指定子ですが、量指定子にはいくつかの種類がありました。

- 貪欲な量指定子（最長一致）
- 懶惰な量指定子（最短一致）
- 強欲な量指定子（絶対最大量指定）

このような量指定子における性質の違いも、選択ポイントの積み方という観点から説明することができます。

例えば最長一致、つまり`/*/*/+`は、短いものから先に選択ポイントに積み上げます。こうすることで上が長いものとなり。結果として最長一致になります。また、最短一致はその逆を行うので、上が短いものとなり、最短一致となります。

そして絶対量指定子はそもそも選択ポイントを積み上げません。この選択ポイントを蓄積しないという特徴のメリットが見えたかと思います。つまり絶対最大量指定子であれば、量指定子を量指定しても大丈夫だと言えます。

ここで前号の記事を見ると、最長一致であればマッチする文字列が、絶対最大量指定子ではマッチしなくなる可能性があると示唆しています。もちろんその通りで、例えば、「donarudo said

*13 先程紹介した NFA というタイプのエンジンです。egrep に搭載されているのは恐らく別のエンジンなのか、必ずしも左を優先するわけではありません。

"ranranru!!" という文章に、*/donarudo said ".+?/"* という正規表現を与えると失敗てしまいます。これは、サブパターン .++/ が 「ranranru!!」 と、最後の 「」 を掴んで離さず、結果としてダブルクオートの閉じ *"/"* にマッチする物がなくなり失敗するわけです。もちろん、最長一致に改めればバックトラックによって、「」 が .+/ から解放されマッチングに成功します。

こうしてみると、ただ適当に量指定子を絶対最大量指定子に書き換えるのは良くなさそうです。どのような場合なら、絶対最大量指定子に書き直せるのでしょうか。

それは、絶対最大量指定子が量指定するサブパターンの集合と、その直後のサブパターンの集合の間に共通部分がなければ良いのです。抽象的な説明ですが、例えば、*donarudo* の話す文字列（クオートの中）にはクオートが含まれない、つまり 「*donarudo said ""Makudonarudo* is better than "Kadofuji""」 というような文章は考慮せず、「*donarudo said "Makudonarudo is better than Kadofuji"*」 というような文書だけ考慮するとした場合を考えてみます。すると、先程 */donarudo said ".+?/"* を使っていましたが、「」 は喋らないとわかったので、*/donarudo said "[^"]+?/"* で良いとなります。

これを絶対最大量指定子に置き換えてみると、*/donarudo said "[^"]++?/"* となります。これは絶対最大量指定子で量指定しているサブパターン *[^"]/* と、その直後に使われているサブパターン *"/"* の間に共通な文字集合は存在しません。なのでこれは絶対最大量指定子に置き換えることが可能です。

どうしてこのようになるのかというと、量指定子はバックトラックを行っても、その文字集合から選ばれた文字の数しか操作しないからです。はじめから集合の外にあるので失敗しているというのに、いくら数を変えたところでマッチするはずがないので、ここは選択ポイントを積み上げるだけ無駄ということです。

クオートで囲まれた文字列の話にもこれが当てはまります。クオートの中の文字列は二種類で、それぞれ、次のようにになります。

- エスケープされた文字
- クオートでもエスケープされた文字でもない文字列

しかし、現在対象にしているのは選択の直後にある量指定子です。選択の直後にある量指定子に適用して良いのかというと疑問が残ります。

選択という状態であると、直後にあるサブパターンがどのようになるのか見当がつきにくいので、一度擬似的に展開してみます。

```
"/ ([^"]+)* \"/ ([^"]+)* \"/ ([^"]+)* \"/ ([^"]+)* \"/ ..... )+ '/x
```

これも選択が */** を使うことで、存在しない可能性を維持出来ています。これを見て分かるかと思いますが、この二つのサブパターンの直後に来る可能性があるものは次の二つです。

- */[^"]+/*の場合

*\\"/*かクオート

- *\\"/*の場合

*/[^"]+/*かクオート

くどいですが、これら二つのサブパターンは共通部分を持たないですし、クオートも含まないので、これらの直後には絶対に共通部分を持たないサブパターンが出現すると考えられます。

また、*/[^"]+/*の量指定子も、同様に直後は共通部分を持たないサブパターンが続くので、絶対最大量指定子に切り替えが可能です。

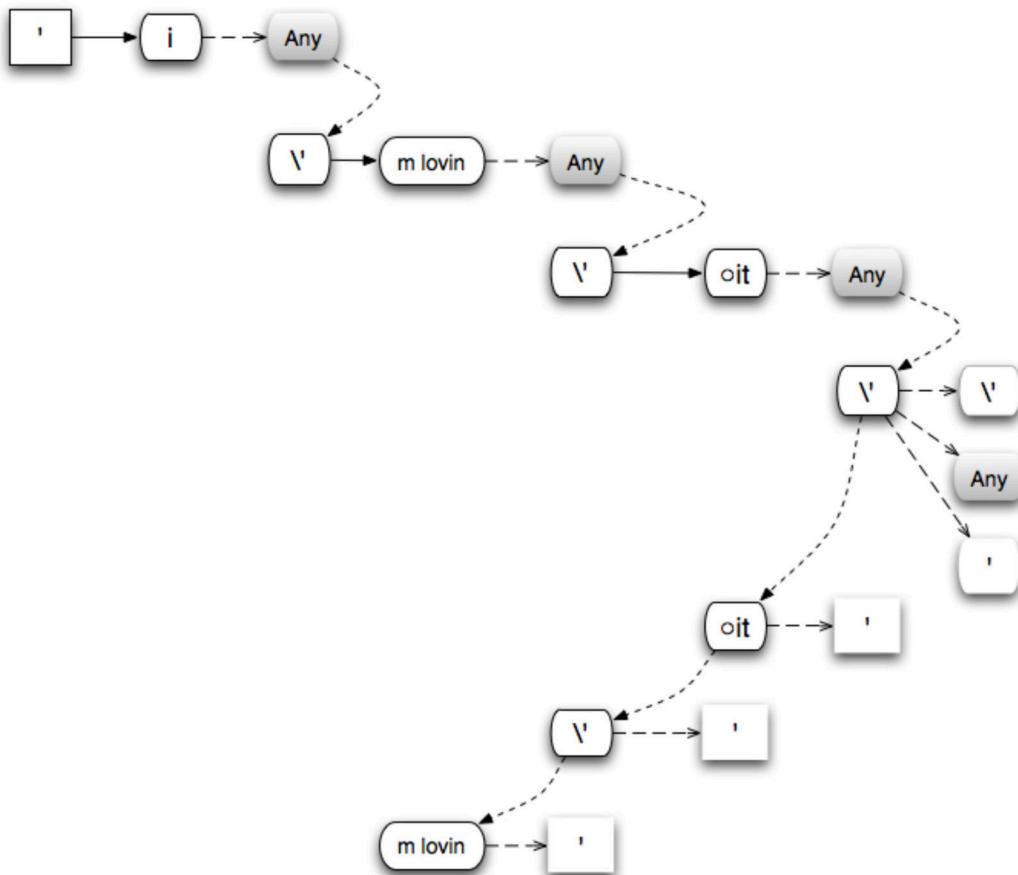
よって、*/([^\"]+\"+)*\\"/* は */([^\"]++\"+)*\\"/* と書き換え可能です。

regex I

とりあえず最適化

ひとまず、先程紹介した次の二つの最適化を施した正規表現を使って作った図を見てみます。

- 選択の順番変更
- 絶対最大量指定子の採用



これだけで最初の正規表現と同じ失敗報告になります。

さらなる最適化とループ展開

```
// ((([^yy']+)* yy* ([^yy']+)* yy* ([^yy']+)* yy* ([^yy']+)* yy* ..... )+ '/x'*14
```

という先程の記述を見て、勘の良い人なら、選択を使わなくとも出来そうだと考えたかもしれません。まさにその通りです。このように選択を量指定子で繰り返す正規表現から、選択を使わない形に改める方法を**ループ展開**と呼び、一般的には次のように行います。

*14 /x修飾子は正規表現中のスペースを無視させるものです。

/開き 一般^{**} (特殊 一般^{**})^{*} 閉じ/x

という形になります。この一般と特殊とは何だ、という感じですが、これらは元の選択に含まれるサブパターンです。今回の場合は、次の二つとなります。

- 一般 エスケープされてないクオート以外の文字列
- 特殊 エスケープされた文字

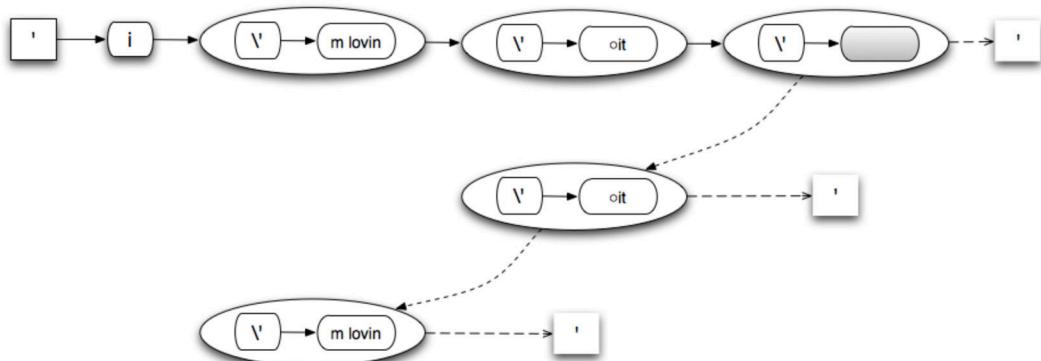
また、これら一般と特殊はそれぞれ共通部分を持たない集合でなければなりません。なぜなら量指定子の中に量指定子が出現しているので、絶対最大量指定子で処理する必要があるからです。

しかし、長い最適化の道のりで、今回のケースでは一般と特殊が共通部分を持たないと判明しているので、絶対最大量指定子を使って、次のように表せます。

/開き 一般^{**+} (特殊 一般^{**+})^{*} 閉じ/x

最後の部分が絶対最大量指定子にならないのは、特殊だけが連続するケースが考えられるからです。

この規則に今まで頑張ってきたクオートに挟まれた文字列を処理する正規表現/([^\W]+|\W)*+/を当てはめると、/[^W]*+(\$|[^\W])*/となります。これで「'I'm lovin' it'」を処理した図がこれです。



すいぶんシンプルになりました！

終わりに

だいぶしんどい作業だったかと思いますが、今回は序論ということで、Perl の意味不明なバックトラック制御記号は使いませんでした。

ここまで読む根性のある方なら、きっと正規表現を最適化出来ると思うので、頑張りましょう。

この記事を読んで、最適化の結果お化けのようになった正規表現を扱いたくないのであれば、あきらめてパーサを書くべきかもしれません。

次回は再帰構造を持つ正規表現について扱うつもりです。

1. こんにちラブライブ!

お久しラブライブ! ラブライブ! と言えば、前に C++だかなんだかの記事を書いた時に、桃ペクチンゼリーから出てくるアタリ券をください、などとシツレイなことを書いていました(詳しくはバックナンバー¹ を参照のこと)。しかしあの後も秋葉原に執拗に通い購入を続け、無事にアタリ券を引き当て、ラブライブ!グッズを入手するに至りましたので、ここに報告します。

さて、本記事が皆様の目に触れる頃には、ラブライブ! 1st ライブのチケット先行予約² の結果が出てる筈です。それはつまり生か死かを分かつ……。

好きなアーティストやアイドルのライブには、矢張り参加したいものです。ぼくは何時もならば、出来れば参加したいという程度の気持ちだったりするわけですが、ラブライブ! のライブには是が非でも参加したいものです。しかし、是が非でもと言った所で出来る事は殆ど無く、人事も尽くせず天命を待つしか無いというのは心細いワケで……。

はあ、もし落選していたら人間を保てるかどうかウッ。人間を保てるかどうかという話で思い出しましたが、今回は正規表現やらのお話です。

正規表現の記事を書くのが流行っているそうなので、今回も皆様にそれとなく、ラブライブ! の文字列を覚えてもらうのが目的です。

2. 正規表現エンジンを作ろう

さて、本記事を読んでいる皆様は日頃使っている言語の「正規表現ライブラリ」や「正規表現モジュール」を使った事がある筈です。本記事は、そんなカチグミの皆様のためのものではありません。

そんなライブラリがある言語使ったことないと答えた方々と一緒に、今回は、「ある文字列が、与えられた正規表現で受理されるかどうか」を調べるプログラムを作っていくたいと思います。

2.1 どんなものが「正規表現」?

例えば、

- /minamikotori/ であっ!! ことりちゃん³ だ!!! 文字列"minamikotori"を受理する正規表現。は～ことりちゃん……
- /(minami.*)&(.*kotori)/ で文字列"minami 適当な文字列 kotori"を受理する正規表現
- /(0|...|9)*(0|2|4|6|8)/ で文字列を数としてみた時、偶数になるものだけを受理する正規表現

ことりちゃんの事を考えていると埒があかないでの、ウィキピーディアに頼ってみましょう。
Regular expression⁴ の項目によると、

- Boolean or 例. gray|grey で、gray か grey を受理
- Grouping 例. gr(a|e)y で、gray か grey を受理
- Quantification 量化子
 - "?" 直前の正規表現が 0 回もしくは 1 回出現する。例. colou?r は color も colour も受理

*1 http://www.word-ac.net/?page_id=569

*2 <http://gs.dengeki.com/lovelive/news/?p=1677>

*3 南ことり: <http://gs.dengeki.com/lovelive/character/chara03.html>

*4 http://en.wikipedia.org/wiki/Regular_expression

- " $**$ " 直前の正規表現が 0 回以上出現する。例. $ab*c$ は $ac, abc, abbc, abbbc, \dots$ を受理
- "+" 直前の正規表現が 1 回以上出現する。例. $ab+c$ は $abc, abbc, abbbc, \dots$ を受理。しかし ac は受理しない！

というこれらの演算子を用いて、正規表現を組み合わせて新たに正規表現を作っていくようです。
詳しく定義に立ち入る前に、正規表現の雰囲気は最早伝わってしまう気がします。

しかし、今回の目的である、正規表現が与えられた文字列を受理するかどうか判定するプログラムを書く為には、何が正規表現であるかをきちんとさせておく(= 定義する)必要があります。とはいっても、ぼくが全く新しく定義を与えるわけではなく、今回はテキスト^{*5}を元に与えることします。

1.2 正規表現の定義

- ϵ は正規表現
- \emptyset は正規表現
- 1 文字の英数字は正規表現
- P と Q が正規表現ならば、 PQ は正規表現
- P が正規表現ならば、 (P) は正規表現
- P と Q が正規表現ならば、 $f(P,Q)$ は正規表現(ただし f は論理演算とする)
- P が正規表現ならば、 P^* は正規表現
- 以上を有限回繰り返して得られるものだけを正規表現とする

いわゆる下からの定義、と言いますか帰納的な定義になっています。

2.3 正規表現の意味

正規表現 R が文字列 s を受理するというのは、正規表現 R の受理する文字列全体を集合として考えた時、 $s \in R$ で表せます。この考え方を用いて

- ϵ は空文字列だけを受理。すなわち、 $\epsilon = \{ \text{""} \}$
- \emptyset はどんな文字列も受理しない。すなわち、 $\emptyset = \{ \}$
- 英数字 c は、長さ 1 の文字列 c だけを受理。すなわち、 $c = \{ "c" \}$
- PQ は、 P で受理される文字列に Q で受理される文字列を結合した文字列だけを受理する。すなわち、 $PQ = \{ pq \mid p \in P, q \in Q \}$
- (P) は、grouping を行うだけで受理する文字列は元の P と何ら変わりない。すなわち、 $(P) = \{ p \mid p \in P \}$
- $f(P,Q)$ は、 P で受理される文字列と Q で受理される文字列とで論理演算を真にするものだけを受理する。すなわち、 $f(P,Q) = \{ s \mid s \in P \wedge f(s) \in Q \}$ 。例えば、 $P \& Q$ (論理積) は、 P と Q 両方で受理される文字列だけが受理される正規表現となり、 $P \mid Q$ (論理和) は、 P か Q のどちらかで受理される文字列だけが受理される正規表現となる
- P^* は、 $\epsilon \cup P \cup PP \cup PPP \dots$ を表す。正確には $P^* = \bigcup_{0 \leq i}^{\infty} P^i$ と定義される。ここで、 P^i は $P^i = P, P^2 = PP, \dots$ とする。特に $P^0 = \epsilon$ であるとする。

*5 オートマトン言語理論 計算論 I

http://www.saiensu.co.jp/?page=book_details&ISBN=ISBN4-7819-1026-2

正規表現で微分

以上のように、正規表現の意味する所を**文字列の集合**として考えると分かり易いです。

でもこの定義を採用すると、量化子のうち、"?"と"+"が扱われていないような……??しかしこれらは、上の定義を用いて簡単に定義することが出来ます。多分。

$$P? = P \mid \epsilon, P+ = P(P^*)$$

うーん、これでなんとかなりそうですね。

さて、このように定義した時に目的は次のように言い直すことが出来ます。

正規表現 L が与えられているとする。この時、入力文字列 str が $str \in L$ となるか $str \notin L$ となるか判別するプログラムを書け。

今こそ、この問題に取り組むべき時です。

2.4 その前に

そのようなプログラムを書くと言いましたが、もう少しプログラムについて注文をつけたいと思います。

というのは、例えばどんな入力に対しても無限ループして、結果を返さずにダンマリというプログラムでは使い物になりませんし、ここまで定義して来た意味がありません。

ですから、次の 2 つを要求します。

- 正規表現 L が文字列 str を受理するならば、プログラムも**受理する**という結果を返す
 - 正規表現 L が文字列 str を受理しないならば、プログラムも**受理しない**という結果を返す
- この 2 つです。これらの要求を満たそうとするならば、取りあえずまともなプログラムになりそうです。

そして要求を良く読むと、正規表現が文字列を受理するかどうかで場合分けされています。ではそもそも、文字列の受理判定というのは決定可能 (decidable) な問題、すなわち受理するか受理しないかという返答を必ず有限時間で出せるものなのでしょうか?

これは yes です。正規表現から NFA や DFA などのオートマトンを作り、それを用いて、文字列を受理するかどうかを有限時間で判定出来ます。詳しくは先に挙げたテキスト⁶ にも書かれておりますので、興味のある方は是非読んでみてください。

「だったら!! その従来の方法を用いれば良いのでは……?」はい。確かにその通りです。しかしそのやり方の場合、1 つの記事で知識を仮定せずに説明を試みると大変面倒くさいので、今回は別のアプローチを採用しようというワケです(逆に言えば今回は十分 1 つの記事で収まる範囲だということです!!)。

いよいよ次節でそのアプローチが明らかになりますが、ここで皆さん自身、どのようにすれば受理判定を行うプログラムが書けそうなのかを考えてみてください。

2.5 正規表現の微分

次のアイディアを考察してみましょう。

入力文字列 $s = c_1 c_2 \dots c_n$ とした時に、正規表現 L が s を受理するかどうかは、 $c_2 \dots c_n$ が

$\{ x \mid c_1 x \in L \}$ に含まれるかどうかを調べれば良い。すなわち、 L のうちで先頭文字が c_1 となっている部分集合に $c_2 \dots c_n$ が含まれるかどうかで分かる。このように続けていくと、空文字列が受理されるかどうかという所に帰着される。

入力文字列の前から消費していくので、これでまあ有限時間で計算出来そうな雰囲気はあります。

*6 オートマトン 言語理論 計算論 I

さて、上で出て来た" $\{ x \mid c_1x \in L \}$ "ですが、これこそが正規表現 L の x についての微分と呼ばれるものなのです。

Brzozowski's derivative operator. 長さ 1 の文字列 c についての正規表現 L の微分 $D_c(L)$ を

$$D_c(L) = \{ x \mid cx \in L \}$$

このように定義する。

定義名の通りですが、正規表現に対する微分という手法を導入したのは Janusz A. Brzozowski で、具体的には 1964 年の氏の論文 **Derivatives of Regular Expressions**⁷ における話です。

いやいや……、この何が微分なんですか？ という質問はごもっともなのですが、これに対する回答はもう少し後で。

それはそうと、凡そ必要なものは揃っていて、上のアイディアを実装すればそれでもう終わり、という雰囲気ですが、実はそう上手くは行きません。

微分の語を使ってアイディアを述べ直せば、以下のようになります。

入力文字列 $s = c_1c_2\dots c_n$ とした時で、 $s \in L$ かどうかを調べるのに、 $c_2\dots c_n \in D_{c_1}(L)$

を調べることにする。最終的に、空文字列が正規表現で受理されるかどうかに帰着される。ここで具体例を考えてみましょう。正規表現 $L = (ab)^*$ とした時に、文字列 "ab" が受理されるかどうかです。これに従えば、"ab" $\in L$ を言う為に "b" $\in D_a(L)$ を言う。さらにその為に "" $\in D_b(D_a(L))$ を調べる、という事に他ならないわけですが、太文字の部分に注目してください。今一度思い出すべき事実は、微分を正規表現に対して定義したということです。つまり $D_b(D_a(L))$ とする為には、 $D_a(L)$ が正規表現でなくてはなりません。しかし、これが正規表現に成っているかどうかは(今のところ)不明です。これこそが問題なのです。

微分操作に対してどのようなことが求められているのかと言いますと、

任意の正規表現 L と、長さ 1 の文字列 c に対して、 $D_c(L)$ が再び正規表現になる。

すなわち、正規表現は微分において閉じていることを示す事が出来れば十分です。言葉を変えれば、 $\{ x \mid cx \in L \} = M$ となるような正規表現 M を常に構成出来るようであれば良いわけです。

さて、この事は既に Brzozowski によって示されています。

2.6 微分はstructurally recursiveに再定義可能

先立って、補助関数 $\delta(L)$ を定義します。 $\delta(L)$ は、 L が空文字列を受理するならば ϵ を、受理しないならば \emptyset を返す関数です。次のように定義出来ます。

$$\delta(\epsilon) = \epsilon$$

$$\delta(\emptyset) = \emptyset$$

$$\delta(c) = \emptyset$$

$$\delta(P^*) = \epsilon$$

$$\delta(PQ) = \delta(P) \mid \delta(Q)$$

$$\delta(f(P,Q)) = \delta(P) \cdot f \cdot \delta(Q)$$

δ 関数を用いて、微分を再定義します。

*7 <http://dl.acm.org/citation.cfm?id=321249>

正規表現で微分

$$\begin{aligned}D_c(\varepsilon) &= \emptyset \\D_c(\emptyset) &= \emptyset \\D_c(c) &= \varepsilon \\D_c(c') &= \emptyset \text{ (if } c \neq c') \\D_c(P^*) &= (D_c(P))(P^*) \\D_c(PQ) &= (D_c(P)Q) \mid (\delta(P)D_c(Q)) \\D_c(f(P,Q)) &= f(D_c(P), D_c(Q))\end{aligned}$$

どれもこれも、受理可能文字列から、先頭の文字 c を削り取ったものを返すという元々の定義は保存出来ている気がします。とりわけ PQ が複雑な定義になっているので、これを少し考えてみます。いや、もう少し具体的には $\delta(P)$ というものが何故混ざっているのか、ですね。ここで重要なことは、 P が空文字列を受理するかどうか、です。これで場合分けすると

- P が空文字列を受理する(すなわち、 $\delta(P) = \varepsilon$)場合
 - $D_c(PQ) = (D_c(P)Q) \mid D_c(Q)$
- P が空文字列を受理しない(すなわち、 $\delta(P) \neq \varepsilon$)場合
 - $D_c(PQ) = (D_c(P)Q)$

まあなんとなくは分かりますね。なんとなくで良いです。

さてさて……。フーム……、これは解析における微分と多少似ている気がしますね!! 多少似てれば似てる。やっぱり微分なんですよこれは。

再定義と書くと語弊がありますが、この等式が成り立つというステートメントが **Derivatives of Regular Expressions** の *Theorem 3.1* です。証明もそれ程難しくはない(ただ、 PQ の場合が要注意なのです)ので、各々読んでもらいたいと思います。

ここで重要なことは、まず δ 関数は引数の構造に対して帰納的に定義された再帰関数(structurally recursive function)です。これは、 δ 関数を任意の正規表現に適用しても停止し、結果を返すことを保証します。

この δ 関数を用いて定義される微分操作関数も、structurally recursive な関数になっていますから、同様に任意の正規表現に適用しても停止します。

さて、再三ですが、ここで最初のアイディアを定義しなおしてみます。

文字列 $s = c_1 \dots c_n$ が正規表現 L に受理されるかどうかを調べる為には

$$\begin{aligned}check([],L) &= \delta(L) == \varepsilon \\check(x : xs, L) &= check(xs, D_x(L))\end{aligned}$$

このように定義される関数 $check$ を用いて、 $check(s,L)$ としてみれば良い。

このようになります。最早これはプログラムですね。ということで、出来るだけそのままに Haskell のプログラムとしてみました。

2.7 Haskell

```
data Reg = Eps | Emp | C Char | Star Reg | Con Reg Reg  
| Not Reg | And Reg Reg | Or Reg Reg  
deriving (Show, Eq)
```

```

(<*>) r1 r2 = Con r1 r2
(<+>) r1 r2 = Or r1 r2
(<&>) r1 r2 = And r1 r2

(<&&>) Emp _ = Emp
(<&&>) _ Emp = Emp
(<&&>) _ _ = Eps

(<++>) Eps _ = Eps
(<++>) _ Eps = Eps
(<++>) _ _ = Emp

delta :: Reg -> Reg
delta Eps = Eps
delta Emp = Emp
delta (C _) = Emp
delta (Star _) = Eps
delta (Con r1 r2) = (delta r1) <&&> (delta r2)
delta (Not r) =
  case delta r of
    Emp -> Eps
    Eps -> Emp
delta (And r1 r2) = (delta r1) <&&> (delta r2)
delta (Or r1 r2) = (delta r1) <++> (delta r2)

derive :: Char -> Reg -> Reg
derive c Eps = Emp
derive c Emp = Emp
derive c (C c') | c == c' = Eps
  | otherwise = Emp
derive c (Star r) = (derive c r) <*> (Star r)
derive c (Con r1 r2) = ((derive c r1) <*> r2) <+> ((delta r1) <*> (derive c r2))
derive c (Not r) = Not (derive c r)
derive c (And r1 r2) = And (derive c r1) (derive c r2)
derive c (Or r1 r2) = Or (derive c r1) (derive c r2)

check [] 1 = delta 1 == Eps
check (x:xs) 1 = check xs (derive x 1)

num = (C '0') <+> (C '1') <+> (C '2') <+> (C '3') <+> (C '4') <+>
      (C '5') <+> (C '6') <+> (C '7') <+> (C '8') <+> (C '9')

is_even = (Star num) <*> (C '0' <+> C '2' <+> C '4' <+> C '6' <+> C '8')

```

正規表現で微分

はいおわり!!

3. この後の話

今回は正規表現を微分するということで、Brzozowski の微分演算についての説明と Haskell での実装、ということになりました。正規表現の微分という概念、正規表現が微分で閉じている、そして structurally recursive であるということを用いて、ヤバイカンタンに実装出来たというところが重点。

しかし、話はまだ拡張することが出来ます。これ以上何を微分するのか!? ということになるわけですが、ズバリ文脈自由文法を微分し、更にパーザコンビネータを微分しようというお話があります。

はい、**Yacc is dead**^{*8} ですね。そんなナンデモカンデモ微分して一体何をするつもりなのか、というと任意の文脈自由文法を扱えるようなパーザを簡単に実装しちゃおうという話なのです。

勿論、任意の文脈自由文法を扱うようなアルゴリズムとして CYK アルゴリズム^{*9} などはあります、微分に基づくアプローチだと更に実装し易くなり、色々な言語で実装出来るようになる、というのがウリです。そうなると、本来パージングをした方が楽なシーンにもかかわらず、化け物のような正規表現を分けもわからずコピペして使っていた人たちでも、適切にパーザを用いてプログラムが書けるようになりますよね!! 嬉しい!! ということです。非常に面白く、かつ実用的な話なので、**Yacc is dead** も是非読んでみてください。

*8<http://matt.might.net/articles/parsing-with-derivatives/>

*9http://en.wikipedia.org/wiki/CYK_algorithm

4. おわりに(もといラブライブ! 情報)



(Amazon.co.jp^{*10} より画像をお借りしました)

おお、見よ！ グレースケールになると、異常なまでのかわいさが抑えられ、直視可能になるんですねえ……。

12月14日にラブライブ！から、南ことりさんソロアルバムの「ことり Lovin' you」が発売されます。2500円!! かわいい!! これまでのラブライブ！で、ことりさんが歌った曲のソロバージョンのみが収録されている、なんと言って良いやら……。まあとにかくかわいいですし皆様もことりちゃんをウツ

*10 <http://www.amazon.co.jp/dp/B005NJXCGM/>

情報科学類誌 WORD 読者アンケート

文・題字 編集部 ふあい

■超速報

COJT グッズ完売！

■あいさつ

みなさん、おはこんばんちは^{*1}。4回目の集計となる今回はなんと**16人**の方から回答をいただきました！常連の方から初めての方まで、ありがとうございます。さて、超速報にもあります通り、第1回のアンケートから配っていた粗品のCOJT グッズ^{*2}が遂に全部捌けました！そろそろ COJT グッズの紹介ネタが尽きてきたので非常に助かりました。重ねてお礼申し上げます。

■今回のステキな粗品

・シェフのきまぐれ粗品

COJT グッズは無くなってしまいましたが、言えば何か~~が~~が出てくるかも知れません。度胸のある方はアンケートの回答を持って WORD 編集部室(3C 棟 2 階、ラウンジ横)までおいで下さい。

時間が無い^{*3}ので、ちゃっちゃと集計へ参りましょう。

*1 おはこんばんちは：おはよう+こんばんは+こんにちは

*2 COJT グッズ：情報科学類および情報メディア創成学類向けの授業として、21世紀の太陽として知られる親愛なる徳永先生の偉大なる現地指導により筑波大学に爆誕した「組み込み技術キャンパス OJT」に関連したグッズ。徳永先生の道楽で作られたという説が有力である。相当量のグッズがばらまかれたため、情報科学類の授業で COJT グッズ所持者を見つけるのは、韓国でキムさんを探すよりも楽だと言われている。色々な経緯があって我らが WORD 編集部にも 10 セットくらい着弾した。

*3 時間が無い：この記事は筆者が締め切りを勘違いしていた為、締め切り当日になって書かれました。

■アンケート集計

◆ Q1:所属を教えて下さい

- ・情報科学類：2人
- ・応用理工学類：1人
- ・社会工学類：1人
- ・工シス学類：1人
- ・コンピュータサイエンス専攻：1人
- ・生物資源科学専攻：1人
- ・iit 学類 master 専攻：1人
- ・社シス専攻：1人
- ・情科学類：1人
- ・隣人部：1人
- ・BAKA 専攻：1人
- ・理工学群社会学類政治学専攻、社工はまやかし。：1人
- ・知識(ry 学類 正式名称でお願いします。間違えたら晒します。：1人
- ・ヒヤッホーワクワク工学研究科 ウフフ専攻：1人
- ・その他の欄が長くなったことに驚きを隠せない第五学群三次元侵略学類：1人

書いてあったとおりに集計しました。かろうじて情報科学類生が一番多いのが分かりますね。個人的にヒヤッホーワクワク工学研究科という響きが好きです。日夜研究に追われながらレッドブル漬けで眼気を吹き飛ばして「ヒヤッホー」と叫ぶ研究員の姿が目に浮かびます。

◆ Q2:性別を教えて下さい

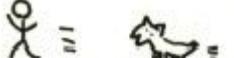
- ・男：7人
- ・女：3人
- ・魔法少女：1人
- ・エルチキ：1人
- ・ともくんのママ：1人
- ・非実在非実在少年：1人
- ・性別とかはスルーの方向で：1人
- ・愛に性別なんて関係ない：1人

ともくんのママから魔法少女まで幅広く読まれる情報科学類誌 WORD を今後ともよろしくお願いします。ところでともくんって誰。

◆ Q3:WORD の公式サイト「WORD Press」(<http://www.word-ac.net>)はご存知でしたか？

- ・はい：3人
- ・もちろん：3人
- ・もちろんりません：1人
- ・パパからきました：1人
- ・答えるまでもない：1人
- ・いいえ：1人
- ・食うかい？：1人

WORD 読者アンケート

- ・もちろん：1人
- ・なん、だと…？：1人
- ・4：1人
- ・ : 1人

ここで、宜しくない読者が数名見受けられました。遺憾の意を表明すると共に、今後の戒めとして無慈悲な晒し上げ措置をとります。

・その1

選択肢が書き換えられているのみならず、WORD の公式サイト「WORD Press」(<http://www.word-ac.net>)を見ていないという有様。誠に遺憾であります。

1:はい
2:もちろんいりません
3:その他(_____)

犯行現場 1

・その2

例外処理として選択肢 3 の「その他」を用意しているのに、さらに選択肢 4 を追加しつつ、しかもその選択肢の説明が全くありません。これでは WORD の公式サイト「WORD Press」(<http://www.word-ac.net>)を見ているのかどうなのか全く分からず、誠に遺憾であります。

1:はい
2:もちろん
3:その他(_____)

犯行現場 2

・その3

日本語で書け



◆ Q4:良かったと思う記事があれば教えて下さい

前号のタイトル一覧は以下の通りです。

- 1.表紙
- 2.目次
- 3.真夏の夜の死闘～東日本縦断レース
- 4.マニアック正規表現 序論
- 5.GNU screen から tmux へ
- 6.本格的♂イタリアン
- 7.全自動ふあぼ bot プロジェクト
- 8.情報科学類誌 WORD 読者アンケート
- 9.編集後記
- 10.裏表紙

- ・2.目次：1人
- ・3.真夏の夜の死闘～東日本横断レース～：10人
- ・4.マニアック正規表現 序論：3人
- ・5.GNU screen から tmux へ：1人
- ・6.本格的♂イタリアン：3人
- ・7.全自動ふあぼ bot プロジェクト：3人
- ・8.情報科学類誌 WORD 読者アンケート：3人

東日本横断レースが圧倒的ですね。18 きっと愛用者として私も参加したかったのですが、都合がつかなかったのが残念です。車組は今ココなう！で、電車組は時刻表で追いかけつつ、ネット

経由でエア参加しました。

それでは、回答欄晒し上げコーナーへ参ります。

3. チャイニーズチキンバーガーを食べたくなりました。

(ヒヤッホーワクワク工学研究科ウフフ専攻 ともくんのママさん)
函館のラッキーピエロの話ですね。私からは「洞爺湖サミットバーガー」をおすすめします。何がどうサミットなのか全然わかりませんが、美味しいのは確かです。

サキは俺の嫁

(情科学類 IMAGINE THE FUTURE.さん^{*4})
結局どの記事の事なのか全然わからなかつたけど、小麦ちゃんじゃないからどうでもいいか。

8: 相変わらず激しい。そして、芝とは一体…! PS. バクラヴァでした(誤植エ…!) 猛良くんが大好きなお菓子と覚えれば間違いないよみんな!

6: イタリアンちゃん tr tr したい。 4: わかりやすかった

(その他の欄が長くなつたことに驚きを隠せない第五学群三次元侵略学類 多加枝 鉢見さん)
tr tr とは一体…? ところで私もイタリアンを食べましたが、一番のオススメは「和風きのこイタリアン」です。イタリアンの基本は焼きそば+パスタソース(←イタリアヘンな要素)なのですが、パスタソースが和風きのこあんかけなので、もはやどこがイタリアンなのか分かりません。麺もソースも日本に帰化しています。味は美味しいので新潟に行った際は是非!

3. エクストリームスポーツ、いいですよね。
6. 3も含め、毎度の事ながら行動力に感動。文章がツボだった。

(応用理工学類 Yellow 13 さん)

身近なエクストリームスポーツの定番と言えば、エクストリーム教室移動ですが、つくば駅に22:40に到着する区間快速からつくばセンターを22:40に発車する循環バス(終バス)へ乗り換える競技は何とかなりませんかね。私の予想では、バスがつくばセンターの信号に引っかかれば、つくば駅から直接吾妻小学校停留所まで走る事により乗れるかもしれないと思っているので、誰か成功したら教えてください。そして関鉄バスはあと**10分**くらい待て^{*5}。

*4 IMAGINE THE FUTURE.さん: アンケートの名前欄がNULLだった回答は、IMAGINE THE FUTURE.さんとして掲載しています

*5 便利な高速バスつくば号をご利用下さい。by 関東鉄道バス

WORD 読者アンケート

真夏の夜の満夢 死闘
理由は無い

(iit 学類 master 専攻 とりすーぷさん)

あのさあ……

3. 鉄道好きにはたまらない内容。 まんべ君かわいいよ まんべ君。

(理工学群社会学類政治学専攻、社工はまやかし。 Temachine 3104 さん)

まんべさん……

4 : Vim の正規表現についてきちんと書いてあった。

5 : 一通り tmux について知れてよかったです。使うかはまた別の話だが…。

(コンピュータサイエンス専攻 hogehoge さん)

正規表現の記事を書いた吉村君はまだ 1 年生ですが、キッチリした記事が書けていてこれから記事も期待できますね。私はこれからも何か書く機会があれば、こんな感じのヘロヘロな記事を書くつもりでいます。

3 学類誌っぽい

7 帰ったらフォローするかもしれない

(知識(ry 学類 正式名称をお願いします。間違えたら晒します。 IMAGINE THE FUTURE. さん)

18 きつぶ VS 車のような、時間をかけて変なことをするのは学生っぽくていいですね。私も立派な学生なので、今年の春に 18 きつぶで名古屋日帰り旅行をしました。名古屋には 30 分くらいしか居られなかったので、矢場とん^{*6} 食って即 U ターンです。帰りの 373 系^{*7} で風邪をもらって数日寝込みました。

4. コンピュータリテラシの復習ができました。

6. おいしそう

(情報科学類 あやさん)

リテラシの授業よりも詳しいよ！

*6 矢場とん：味噌かつ店。美味しい。

*7 帰りの 373 系：18 きつぶ旅行で 373 系といえば静岡駅 19:30 発の東海道線普通東京行き(373 系 9両編成)というのは、もうお分かりですよね。

7. fav_bot

時期を同じくして、そしあでもふあぼネタが掲載されており、興味深い。

8. いつも通り面白い

(社会工学類 黒谷駆さん)

手元には7月に発行されたそしあ～る129号しかありませんでしたが、これにもTwitter記事がありましたね。ところでこれの19ページ注釈1行目の「非リアとは、「リア充」とは、リアルが充実した……」という文章は日本語として文法がおかしいですよね。それから、えーっと、まあ、そしあをいじるのはこれくらいにしておいてやろう。

2. 目次 見やすいです

8. アンケート もはやこれがメインに思える

(工システム学類 Elleryさん)

メインっぽく見えるのは、毎回アンケート用紙が入っていて目立つのと、やけにページ数が多いからでしょう。

③今度は西日本編を期待しています

(隣人部 エア友達もQBも見えないんだよねさん)

西日本編をやるとしたら、名古屋周辺と京阪神は新快速様が走っているので鉄道圧勝ですね。

3. 旅記事は好き。今回はスケールが大きくて楽しかった

(情報科学類 IMAGINE THE FUTURE.さん)

ここ最近の旅記事といえば、車と18きっぷが戦ったり、奥多摩の廃線を見たり、うさぎと戯れたり、普通の旅が全く無いですね。まあ普通の旅はWORDには載りませんが。

3. 今度から札幌の実家に帰る際、18きっぷをつかいます。

(社シス専攻 果実系男子さん)

今は18きっぷだけで札幌へ行けますが、北海道新幹線が開業したら江差線の木古内～五稜郭間が廃止になるとか、第三セクターに移行するとか言われていますね。今後どうなるか気になるところです。

WORD 読者アンケート

3. 敬意を表するツ！

(BAKA 専攻 吉井明久さん)

(L*) ▾

3. こーゆー一体はった記事大好きです♥

7. ちょーどえりたんに興味持ったところだったので かなり真剣に読んでました。
(ってかこゆのさらっと作っちゃう工学系ってスゲエ。)えりたんはお姉さんがお嫁さんとしてお迎えします♥♥

(生物資源科学専攻 いちごん。さん)

お嫁さんを大切にしてあげてください。えりたん bot のバス時刻表機能は実に偉大。

◆ Q5: 良くなかったと思う記事があれば教えて下さい

- ・1.表紙：3人
- ・3.真夏の夜の死闘～東日本横断レース～：1人
- ・8.情報科学類誌 WORD 読者アンケート：1人
- ・9.編集後記：1人

表紙に3票はいっていますが、そのうち2つが号名に関するものでした。今回から表紙とは別に号名を選択肢に加えておきましょう。以下、恒例の回答欄晒し上げコーナーです。

Perl6 の記事をもっと書け！ 布教だ！ 洗脳しろ！

(ヒヤッホーワクワク工学研究科ウフフ専攻 ともくんのママさん)
ご要望にお応えして、今号はがつり書きました。

3: 左右で組が分かれていたようですが、▼を鉄道▽を車組とかにした方が時間関係が
わかりやすかったのでは。

(その他の欄が長くなったことに驚きを隠せない第五学群三次元侵略学類 多加枝 錫見さん)
数少ない参考になる意見、ありがとうございます。あまりこういった記事構成は無いのですが、
今後の参考にさせていただきます。

?

(iit 学類 master 専攻 とりすーぷさん)

???

9. クオリティーが下がった気がした

(理工学群社会学類政治学専攻、社工はまやかし。 Temachine 3104 さん)

【業務連絡】WORD 編集部員に告ぐ。編集後記ではっちやけるように。

今回はない。

(コンピュータサイエンス専攻 hogehoge さん)

やったね！

1 一斗缶とは何だったのか

(知識(ry 学類 正式名称でお願いします。間違えたら晒します。 IMAGINE THE FUTURE. さん)

1. どうして一斗缶なんだい？訳が分からないよ。

(BAKA 専攻 吉井明久さん)

毎回、号名はハイテンションになった部員達のノリと勢いで決められます。今回は特にハイテンションだったんでしょう。私ももう覚えてません。

①何で急にスタイリッシュになってんの？萌絵は？痛いコピペは？

まどマギと同じ表紙詐欺ですね。

(隣人部 エア友達も QB も見えないんだよねさん)

何を言ってるでゲソ？表紙も中身もスタイリッシュじやなイカ！



(社シス専攻 果実系男子さん)

何なんでしょう、**実際にコメントに困る**回答が来てしまいました。今度からコメントに困る回答が来たらこの絵を使うことにしましょう。

WORD 読者アンケート

えー…っと、このアンケートってマジメに回答したら負けなんですかね…？

(生物資源科学専攻 いちごん。さん)



◆ Q6:過去の記事に関する感想を教えて下さい

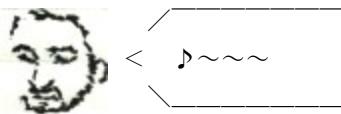
頭を打ってから過去の記事と7の段が思い出せません。

(ヒヤッホーワクワク工学研究科ウフフ専攻 ともくんのママさん)

過去の記事はWORDの公式サイト「WORD Press」(<http://www.word-ac.net>)を、7の段は Wikipedia の구구법のページ (<http://ko.wikipedia.org/wiki/구구법>) を参照してください。

♪～♪～♪～♪

(情科学類 IMAGINE THE FUTURE.さん)



エルチキにポン酢とマヨネーズとこしょうとあおのり かけるとうまいよ

(iit学類 master 専攻 とりすーぷさん)

太りそう。

アメリカンパワーフード食べたい。

割とガチで(`・ω・ `)

(理工学群社会学類政治学専攻、社工はまやかし。 Temachine 3104 さん)

アメリカンパワーフードの記事を参考に、どんどん作って、ぶくぶく太りましょう！

なんかパーティーとバケツがあったのは覚えてる。

(知識(ry 学類 正式名称でお願いします。間違えたら晒します。 IMAGINE THE FUTURE.さん)

MAX プリン(バケツ入り)ですね。一時期 WORD で猛威を振るっていましたが、最近はめっきり作られなくなりました。今は MAX キャラメルの時代です。

昨年のポケモン特集は病気だと思いました。

あれがポケモンの廃人か…。もっとまともな生活を送るべきだと思います。

戦わなきや、現実と！

(コンピュータサイエンス専攻 hogehoge さん)

昨年のポケモン特集を書いた IX 氏いわく、「私は乱数に手を出してないだけマシ」だそうです。

毎回楽しみにしています。

(情報科学類 あやさん)

ありがとうございます。

あんこちゃんあんあん

野中藍の声可愛すぎだろ

(隣人部 エア友達も QB も見えないんだよねさん)

これ、どの記事の感想なんだ。

バトルトードはどうなりました？

(情報科学類 IMAGINE THE FUTURE.さん)

色々ありましたが、今度**供養**します。

フルーツ男子を再開してほしい。

(社シス専攻 果実系男子さん)



過去…というか、今回というかですが、アンケ掲載ページには前回のアンケの設問をちまっと載せといってくれるうれしいです。

(生物資源科学専攻 いちごん。さん)

ご意見ありがとうございます。載せました。喜んでください。

WORD 読者アンケート

◆ Q7:自由記述欄(この記事のメイン)

プログラミング入門 I との戦い
いや、いい

(情科学類 IMAGINE THE FUTURE さん)

プログラミング入門 I でショートコーディングとかして、TA をリンチするのはやめようね☆

葡萄酒さんのファンです！
絶対この人 小説書いてると思う
あと小麦なる人物とアンケートの出す場所が
わからぬ。

(工システム学類 Ellery さん)

彼はなかなかカッチョイ文章を書きますが、小説は書いていないそうです。ただ、普段からかなり小説を読んでいるようですが。

すーぱーそに子かわいい！

(BAKA 専攻 吉井明久さん)

すーぱーそに子がどんなキャラなのか、Wikipedia 先生に聞いてみたときのキャプチャがこちらになります。

すーぱーそに子

すーぱーそに子(すーぱーそにこ・SUPER SONICO)は、ニトロプラスのイメージキャラクターの名称である。音楽CDやフィギュア等の

かわいいですね ^ ^

文字がつ……！小さ
いっ……！印刷に載る
かどうか……！

印刷時につぶれた時用に要点をまとめておくと、「天地無用！」と銀さまと星界シリーズが好きでゴザル」という内容です。

天地無用！といえば PC-FX の「天地無用! 魁皇鬼 FX」というゲームですよね。PC-FX の時点で入手困難ですが何故か手元にあります。いいだろー。

<p>天地無用！が通じるのは嬉しいですね…笑 まさに、ダレホールに了【天地無用】に反応してしまう次生(?)がありましたが。 (同じ様に「乳酸菌」にも反応してしまいます。)</p> <p>天地無用！だと、ノベル版の「魁皇鬼外伝 天地無用 GXP」もオススメです。 美星さんは、(魁皇鬼だと)確率の偏りで、運がいいらしいので、 もうちょっとありますよ！(モノ)は駄目になりますが、</p> <p>星界シリーズの作者森田 浩えさんの短編集「魔の封印が抜けたなら」叢録の「スパイク」が最近のヒットです。なかなかの驚きでしたか。 これに触発されて(驚耐性がでて?)「陰陽了」と「モモを出したか」、 面白がたりです。 合わせて宣伝しています。</p> <p>PS. 「ULTIMO」と、マークシングのミニタス版描き下ろしマンガで 道を踏み外してます一笑。</p>

(応用理工学類 Yellow 13 さん)

1 : 私のスケジュールは null で埋まっています。

2 : Perl6 が広まらない理由としては、ドキュメントが整備されていないのが大きな理由のひとつでしょう。未だに追加仕様が頻繁に発生するため、ドキュメントが書き辛いという背景があります。

3、6 : Perl の優位性といえば、やはり CPAN の存在が上げられます。モジュールが簡単に利用できるのもそうですが、依存関係やテスト状況が一覧できる CPAN のサイト自体が良く出来てると思います。Rubyに対する優位な点として、言語の仕組み自体が非常にシンプルなので、「簡単な事がより簡単に出来る」というのが私の持論です。

4 : はい。

5 : そういった世界があるという事実を発信したいからです。

7 : ですよね。

8 : 卵の孵化作業といった単純作業が出来る人間は、社畜の素質がありますので、それをアピールしてください。

<p>1. WORD 編集部員は日々どんなスケジュールで活動しているのですか？</p> <p>2. Perl6 はなぜ広まらないんですか？ いつもリースされるんですか？</p> <p>3. 私に Perl の魅力を教えてください。</p> <p>4. プログラミング記事？ いいや、もとや。</p> <p>5. ときどき 頭がおかしい 常人には理解できない記事があるのはなぜですか？</p> <p>6. Ruby が最近流行っていますが、Rubyに対する Perl のアドバンテージとは何でしょうか？</p> <p>7. 粗品？ いいません。</p> <p>8. ポケモンを就活に役立てる方法はありますか？</p>
--

(コンピュータサイエンス専攻 hogehoge さん)

WORD 読者アンケート

>小麦ちゃんのばんそうこう貼ったりはがしたりしたい

>小麦ちゃんのばんそうこう貼ったりはがしたりしたい

>小麦ちゃんのばんそうこう貼ったりはがしたりしたい

>**小麦ちゃんのばんそうこう貼ったりはがしたりしたい**

分かってますね！機会があれば是非お話しましょう！

(ヒヤッホーワクワク工学研究科ウフフ専攻 ともくんのママさん)

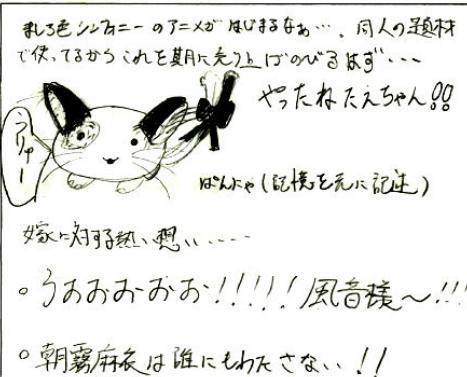
ここ数年はほとんどアニメを見ていないので、アニメの話題についていけません。毎週見てるアニメはサザエさんと黄金バットくらいでしょうか。かろうじてイカ娘の話題ならついていけます。

ところで嫁に対する熱い思いを見ていたら無性に**キヤベツ**が食べたくなりました。どどんまい！

シムーンという、2006年を百合色に染め上げるところを名作があるのですが、これを見ると俺も黒かつては少女だったような気になつてよいです。

あと、ソラ・ソラ・トの1話と13話だけ見て間を想像で補完すると名作を見た気になれるのでよいです。

小麦ちゃんのばんそうこう貼たりはがしたりしたい。
かじこ



(理工学群社会学類政治学専攻、社工はまやかし。 Temachine 3104 さん)

ひー、時間無いので厨房の時に書いた小麦ちゃんで勘弁してください。



小麦ちゃんかわいいな～。

小麦ちゃんって誰だ

自分で描いたイラストを載せて下さい

南の学類誌

「milk」(ミルク) (知識)

「MAST」(マスト) (×創) もよろしくお願ひします

(知識(ry 学類 IMAGINE THE FUTURE.さん)

研究室紹介号を3年生以外に配りたいという話は、編集部内でも挙がっていました。

そんなわけで、今年度は3年生以外の人にも配れるよう、多めに刷ったので、**研究室紹介号が欲しい方は編集部まで来ていただければ差し上げます。**

みんなどんどんもらいやに来てね。

前から感じていましたが、研究室紹介とかあってもいいと思います。
3年の時に3年生に向けてのみ特別号が発行されますが、それではもったいなーかな。
意識が高い人は自分で研究室とか1,2年の頃に行くけど、そんな人は稀有な存在です。
私が早い時期から道路の参考にできるように特徴ある研究室などを取材してちょこちょこ紹介記事を掲載したりとか

乱筆にて失礼

(情報科学類 IMAGINE THE FUTURE.さん)

まとめサイトと違って、このアンケートは今のところ全ての回答を晒し上げています。そして、載せちゃダメチェック欄のマークがチェックかどうか分からぬものは**全部載せます。**

もし回答欄内にアフィリエイトが貼ってあった場合には、面白い物を除いてIMAGINE THE FUTURE.のロゴを貼るなどの対応をとるかもしれません。

2chに書き込んだ「俺のレス(どちらか)」がしばしばまとめサイトに載るのですが、このアゲが冊子に載ったときと同じ様な気持ちになるんですね。アフィックスつけた? 載る? が?

今期のアニメでは「はがなー」が面白いですね。
肉がわいいよ肉。金髪巨乳"ソンデ"しなくて
いままで好きになつたことないのに星奈には
ぶひひひひひひひひひひひひひひ
いいいいいいいいいいいいいい
これが言いようがないです。

アゲ

氏名(PN可): エア友達もQBを見えないんだよね

意見・感想等の自由記述欄を載せられたくない方は、右の欄にチェックをお願いします。□
ご協力ありがとうございました。

(隣人部 エア友達も QB も見えないんだよねさん)

WORD 読者アンケート

とりあえずここに NITRO BOX 置いておきますね



シュー……

(こり) がちいいよ

まじか、は。



NITORI BOX 買えよ

(iit 学類 master 専攻 とりすーふさん)

「私が絵を描くには、このアンケート用紙は狭すぎる」というわけで、アンケート用紙とは別の A4 用紙に描いていただきました！

いやー、かわいいなー。

2学の食堂に関する意見が書かれていますが、私が1年の時に2学食堂で出会った謎の食料についてコメントするには、このスペースは狭すぎるのでまたの機会としましょう。

ところで今度小麦ちゃん描いて下さい！



(生物資源科学専攻 いちごん。さん)

こっちもかわいい！このページは実際に目の保養になりますね。毎回イラストを描いて貰うばかりではアレなので、私もエコハちゃんの絵を描きました！

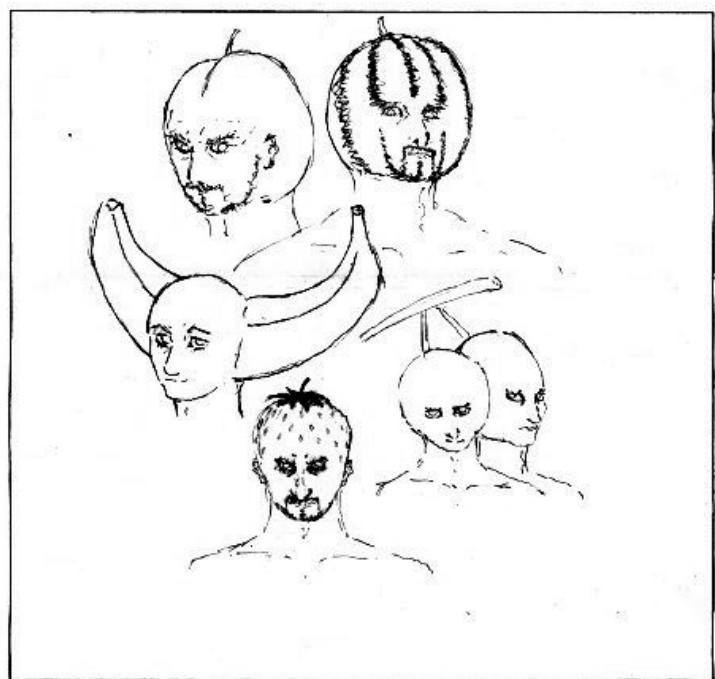
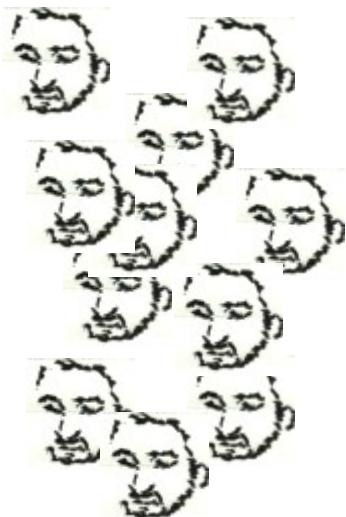


本当は4ヶ月くらい前に描いたんですけどね。



(その他の欄が長くなったことに驚きを隠せない第五学群三次元侵略学類 多加枝 鋸見さん)

実際にコメントに困りますね！そ
うか、こういう時こそ……！



(社シス専攻 果実系男子さん)

★今回のオチ★

何これ怖い！

擬人化って書いてあるけど、全然人じやないし、ポテトも全然ポテトっぽくないし！見ているだけで恐ろしさを感じる！これが地獄先生ぬ～べ～に出てきたら、絶対「オオオオオオオ……」っていう悪霊が出てきたときの効果音付くよ！

何より恐ろしいのはこれがオリジナルじゃないって事だね！

誰だよ原案！



(情報科学類 あやさん)

WORD 読者アンケート

■ここで大切なお知らせがあります。

次回からこの記事の担当が ItosugI 君にかわります。私との違いを以下の表にまとめました。

	ふあい	ItosugI
誕生日	20世紀	20世紀
年齢	20代	20代
出身	関東地方	関東地方
入学年度	2008年度	2008年度
所属	情報科学類	情報科学類
性別	男	男
学年	4年	2年

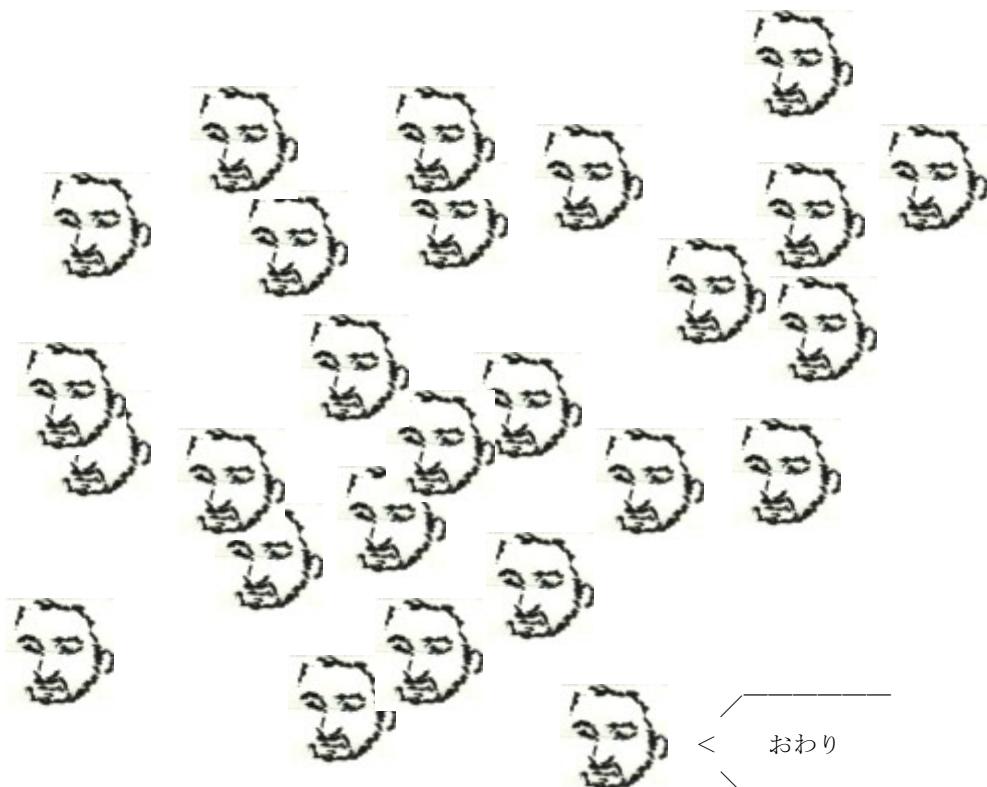
←完全に一致！
←完全に一致！
←完全に一致！
←完全に一致！
←完全に一致！
←完全に一致！
←だいたい一致！

つまり、ほとんど変わりは無いので、ただちに記事の品質には影響いたしません。次回からも「情報科学類誌 WORD 読者アンケート」をよろしくお願いします。アンケートの回収はいままで通り 3C113(3C 棟 1 階計算機室)前、3C205(3C 棟 2 階計算機室)前、情報科学類生ラウンジ(3C 棟 2 階)の 3 カ所で行います。それではさようなら～。

■スペース余っちゃった

困ったな…… いまさらレイアウトを変えるのも面倒くさいし……

そうか！ こういう時は…… !



編集後記

情報科学類誌

WORD

From College of Information Science

WORD燃やすぞ号

発行者
編集長
制作・編集

情報科学類長
中島光夫
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)

2011 年 12 月 16 日 初版第一刷発行
(512 部)