TSNET スクリプト通信

August, 2008　　TSNET

巻頭言 -「TSNET スクリプト通信」第 2 号刊行

盆休みに突入。ようやく「TSNET スクリプト通信」第 2 号の編集に取り掛かる。自分自身の記事が間に合うか、あやうい感じで進んできたのだが、機械伯爵氏、Ｙさ氏の原稿が後押ししてくれて、なんとかまとめることができそうな雰囲気になってきた。実は、巻頭言を書き始めた段階では、まだ原稿は書いていなかったのだがスクリプトは動きはじめていた。「実践実用 Perl」を書いた時のようだ。

創刊号から 3 ヶ月、あっという間に時は過ぎる。僕自身は、オブジェクト指向プログラミングを経て、アクターモデルへの関心から Erlang を動かしてみたり、Rational programming 経由でゲーム理論を調べてみたりしていた。しかし、個人的に実用的なレベルではまだ意味を持つには至らない。

第 2 号(1.2)は、機械伯爵さんのプログラミング入門本の書き方についてのお話、そしてＹささんとくれば AWK スクリプトゲームとなるわけで、例によってコマンドラインで動作する。そして、第 2 号から、特別ゲスト、FGALTS 時代、スクリプト実行環境 Copal で御馴染みだった海鳥さんに「よしおさんとロボ太」シリーズのマンガを掲載させていただくことになった。次に jscripter こと僕の「Wemo をつくる」という自作のススメの計 4 編となった。

表紙には、広島みなと夢花火大会のケータイ写真を並べてみた。夏号ということで・・・次回は僕のケータイの写真じゃなくて画質のよい写真を載せたいものである^^;)

2008 年 8 月 16 日
jscripter

オバケでもわかるプログラミング入門書を……

by.機械伯爵

０．オバケはプログラまないか？

家に居候しているオバケの化太（ばけた）が、午前の本屋巡りから帰ってきた。
未確認飛行物体の分際で、いい身分である（オバケは浮いて移動する）。

化太「機械しゃん、化太はＩＴするぎゃ」

機械「なにをいきなり」

化太「ＩＴはモテモテぎゃ。化太はモテたいぎゃ」

機械「オバケ世界の流行ってのは遅れてるんだな」

化太「人間世界でもトレンドぎゃ」

機械「ちげーよ。ＩＴとか、そろそろ忘れてるぜ、みんな。ま、意味が分かってたとも思えないけど」

化太「ＩＴはモテぬぎゃ？」

機械「オバケモードは知らんよ、俺は。それとも人間にモテたいのか？」

化太「人間はおどかす相手ぎゃ。化太はつるぺたろりぷになオバケ美幼女にモテたいぎゃ」

機械「……オバケの世界は無法地帯だな。まぁ、オバケがオバケをどーしよーと俺の知ったこっちゃないが」

化太「というわけで、ＩＴぎゃ」

機械「ＩＴってなんだよ」

化太「プログラミングぎゃ。ＩＴはコンピュータぎゃ。コンピュータはプログラム無しでは動かんぎゃ。プログラムを制する者はＩＴを制するぎゃ」

機械「明らかにイロイロ間違ってるんだが、もはやツッコむ気にすらならん。んで、入門書とか見てきたのか？」

化太「見てきたぎゃ」

機械「んで？」

化太「難しくてわからんぎゃ」

機械「いったいどんな本を見てきたんだ？」

化太「Ｃ言語とか Java とかの本ぎゃ」

機械「Ｃとか Java の本なら沢山あるから、分かりやすい本もあっただろ？」
化太「オバケには難しいぎゃ」

機械「ネコでも分かる本があるってのに……」

化太「機械しゃんは、ネコ本はわかるぎゃ？」

機械「いや、読んでみたがさっぱりわからんかった」

化太「ネコ以下ぎゃ」

機械「ネコ未満だな、正確には」

化太「ヒトのプライドはどーしたぎゃ？」

機械「所詮九割は機械だからなぁ」

化太「一割の中身は、美人の未亡人を剥製にして飾ることで一杯ぎゃ？」

機械「ふ、若気の至りだな。あのとき脳に大穴を開けられて残りの九割も吹っ飛んだが」

化太「重力の墓場に捨ててきた女の呪いぎゃ」

機械「ストーカーのリアルヤンデレだぞ、あの女はっ！　それに機械化はあの女の趣味だっ！　強制されたとか捨てられたとか、電波な自作自演なんだよ全部っ！」

化太「ま、そーいうことにしとくぎゃ（ふっ）」

機械「で、プログラミングの話はどーなったんだ？」

化太「プログラムを書きたいぎゃ」

機械「で、Ｃや Java では難しいと？」

化太「絵本でも脳が沸騰したぎゃ」

機械「（それって、根本的にプログラミングとか無理じゃねーのか？）で、Ｃや Java 以外は？　Perl とか Ruby とかの本、あるだろ？」

化太「他言語経験が前提ぎゃ」

機械「んなことねーだろ。たしか Ruby でプログラミングに入門する本、あったはずだぞ」

化太「近所の本屋には無かったぎゃ」

機械「amazon.com でも使えよ」

化太「通販はイヤぎゃ」

機械「なら街の大型書店に行けよ」

化太「面倒ぎゃ。大体、プログラミング言語はこんなにたくさんあって、プログラミングの本も本屋に結構置いてあるのに、CやJavaからの入門書しか無いなんておかしいぎゃ。抗議するぎゃっ！」

機械「誰に？」

化太「世間ぎゃ」

機械「そんなこと話題にするほど、世間は暇じゃねーよ。ま、それはそうとして、Visual BasicとかHSPの本とかはどうだ？　結構初心者向けだぞ？」

化太「書いてある通りにやればプログラムは出来上がるぎゃ。でも、自分で何やってるか、イマイチわかりづらいぎゃ」

機械「そんなん、やってるうちに覚えるんだよ」

化太「オバケには向かんぎゃ」

機械「なら、どーしろってんだよ」

化太「機械しゃんが書くぎゃ」

機械「はぁ？」

化太「機械しゃんが、化太のニーズに合う本を書くぎゃ」

機械「無茶を言うな。俺は本業が忙しいんだよ」

化太「でも、ラノベの新人賞に応募する原稿を書く暇はあるぎゃ？」

機械「俺の趣味に口出さないでもらおうか」

化太「萌え系ソフトポルノ書く暇があるなら、哀れなオバケのプログラミング入門者用の本を書くぎゃ」

機械「てめ、ラノベを何だと思ってるんだ？」

化太「半分は、成人用同人誌の直接的表現でも昇華できない少年少女のリビドーの解消用ぎゃ」

機械「煩悩で成仏できないオバケに言われる筋合いはねえ」

化太「オバケはオバケぎゃ。人間は死ねばそれまでぎゃ。オバケが死んだ人間に化けることはあっても、死人が化けることは無いぎゃ。死人は未分解の有機物にすぎぬぎゃ」

機械「うごーっ！　仏様をゴミ扱いしやがってこの罰当たりがぁっ！」

化太「生命を冒涜しまくってる機械化人間に言われたく無いぎゃ。つべこべ言う前に書くぎゃ。でないと、**ガニマタのチビに、機械しゃんが生きてること、バラす**ぎゃ」

機械「……マジ怖いからヤメろ。わかったわかった。じゃ、まずどんな本を書くか、打ち合わせするか」

化太「どうせなら萌え本を書いて大もうけするぎゃ」

機械「それも古いよ」

１．オバケでも読める本？

機械「で、どの程度の本なら読めるんだ、お前？」

化太「１８禁でもＯＫぎゃ」

機械「ふぅん、成人向けね。じゃ高等課程程度の教養はあるもの、と」

化太「こーとーかてーって何ぎゃ？」

機械「高校生が習ってる程度の教養だよ」

化太「無理ぎゃ。オバケにゃ学校も試験もなんにもないぎゃ。高校球児より知識無いぎゃ」

機械「失礼な、野球やってたって勉強できる人はたくさんいるんだぞ。それはさておき、まぁ、義務教育の教養レベルってとこか？」

化太「ぎむきょういくって？」

機械「中学生が習ってる程度の教養だよ」

化太「無理ぎゃ」

機械「おい、成人向けＯＫなんだろ？　一般の成人は、義務教育はちゃんと受けてるはずだぜ」

化太「受けてるのと知ってるのは別ぎゃ。機械しゃんだって、４８都道府県全部は言えんぎゃ」

機械「いいんだよ、どうせしばらくしたら道州制になって減るんだから」

化太「情けない後ろ向きぎゃ」

機械「うるさいうるさい。わかった、小学校卒業レベルでいいんだな」

化太「化太は小学生でもＯＫにゃ。機械しゃんも小学生大ＯＫにゃ」

機械「頼むから、人聞きの悪いことは言わないでくれ。少なくとも現在は、人畜無害なんだから」

化太「近所の小学校から生徒をお持ち帰りしないぎゃ？」

機械「せんわいっ！」

化太「じゃ、地下の秘密監禁調教ルームは何のためにあるぎゃ？」

機械「そんなアヤシゲなもんは無いっ！」

化太「ウソにゃ。地下３０メートルに、確かにあったぎゃ」

機械「そんな深くまで、どーやって行くんだよ？」

化太「入り口は埋まっていたぎゃ」

機械「……」

化太「でも、古いノートに、人間をアレコレした記録が一杯あったぎゃ」

機械「戦時中の人体実験施設かな？　でもそれでも深すぎるなぁ」

化太「去年の夏に、化太と仲間が持ってきて埋めたぎゃ」

機械「……」

化太「いざというときに、機械しゃんを脅迫するネタになるぎゃ。これは秘密ぎゃ」

機械「……」

化太「機械しゃんは心に疚しいトコロがあるから、こういう物的証拠は致命的ぎゃ」

機械「……話が進まんから放置だ。で、教養レベルは小学生の知識程度でいいんだな？」

化太「最初からそう言ってるぎゃ。機械しゃんは物分かりが悪いぎゃ」

機械「誰の為にやってると思ってんだ。まぁいい。んでお前、パソコンは普通に使えるんだな？」

化太「WinならＯＫぎゃ」

機械「ならよし。パソコンを普通に使えんヤツ相手に、プログラミングの本なぞ書きたくないしな」

化太「当然ぎゃ」

機械「じゃ、最初はシェルを使うということで」

化太「シェルって何ぎゃ？」

機械「Windows で言うとこの、コマンドプロンプトってヤツだ」

化太「コマンドプロンプトって何ぎゃ？」

機械「使ったことねーのか？」

化太「ないぎゃ」

機械「（画面でコマンドプロンプトを立ち上げて）コレだぞ、使ったこと、本当にねーのか？」

化太「初めて見るぎゃ」

機械「パソコン、普通に使えてないじゃねーか」

化太「普通は使わんぎゃ。見るにそれ、カビが生えたパソコンの画面ぎゃ」

機械「うるせえ。シェルのＣＵＩは基本だ。プログラミングするってなら、ＣＵＩから始めろ」

化太「ＧＵＩがいいぎゃ」

機械「却下だ。ＧＵＩでは、プログラミングの動作が理解しにくい。まずは入出力がしっかりと見えるＣＵＩプログラミングが基本だ」

化太「しかし、シェルの使い方、知らんぎゃ」

機械「プログラミングに必要な最低限度は覚えるんだな。ま、考えてみれば今のご時世、そのくらいは入門書に書いてないとまずいかもな」

化太「書くぎゃ」

機械「そのかわり覚えろよ」

化太「善処するぎゃ」

機械「あと、学習に使うパソコンは……オレの古でいいな」

化太「文句は言わないぎゃ。しかし、ネットが繋がってないぎゃ」

機械「いらんよ。プログラミング練習用のパソコンは、セキュリティをある程度外すこともあるから、ネットに繋がっていないほうが好都合かもしれん」

化太「開発環境が手に入らんぎゃ」

機械「ネットカフェかどっかで落としてきて入れりゃいいだろ？　どっちにしろお前はオレのをやるから問題あるまい」

化太「ネットカフェに持ってくストレージは、アダルトビデオのネット配信データを入れるので一杯

ぎゃ」

機械「もう一本持っていけよ。それにお前には要らんとゆーておる」

化太「機械しゃんのコレクションは化太の趣味と少し違うぎゃ」

機械「開発環境の話だ」

化太「紛らわしいぎゃ」

機械「紛らわしくしてるのはお前だ」

化太「責任転嫁は良くないぎゃ」

機械「その言葉、そっくりそのまま返してやろう。真面目にやる気が無いなら、俺は降りるぞ」

化太「真面目になるぎゃ」

機械「……よく考えたら、理工系学生のための入門書とかでいいのか？　なら何冊か俺の本棚にもあるぞ？」

化太「（ぱらぱらと内容を見て）あかんぎゃ。わからんぎゃ。人間の言葉じゃないぎゃ」

機械「つーてもコレで、世の中の学生は勉強してるんだが」

化太「オバケは数学が苦手だぎゃ」

機械「お前が苦手なだけだろ？　それにそんなに難しい数学、出てないぞ」

化太「解の公式はダメぎゃ。運動の法則もダメぎゃ」

機械「……解の公式はともかく、運動の法則はただの掛け算だろうに」

化太「物理はダメぎゃ、じんましんが出るぎゃ」

機械「数学使わずにプログラムって、すげぇ難しいぞ」

化太「算数はできるぎゃ」

機械「わかった。算数レベルの問題限定、だな。で、ソフトウェアを作りたいのではなく、あくまでプログラミングを学びたいと」

化太「ＰＳＰのゲームとか作りたいぎゃ」

機械「却下。特定ハードのプログラミングは、プログラミングがまずできるようになってからだ」

化太「ＰＳＰは無理ぎゃ？」

機械「プログラミングを理解せずに、ならＯＫだが？」

化太「不許可ぎゃ」

機械「なら最初から欲を出すな。ハードウェアコントロールは難しいんだ」

２．基本ってなに？

機械「じゃ、次は内容だな。なんといっても基本は……」

化太「オブ脳ぎゃ」

機械「微妙に古いネタだな。それに、オブジェクト指向プログラミングに触れるとしても、最後の最後だぞ」

化太「関数型ぎゃ」

機械「やってもいいが、使える言語は限られるぞ？　それに数学ばりばりな人に向いてるパラダイムだからなぁ」

化太「なら普通でいいぎゃ」

機械「そうそう、普通が一番」

化太「普通って言うなぎゃ」

機械「はい、お約束。それはさておき、やはり構造化定理が基本だろうなぁ」

化太「何ぎゃ？」

機械「ダイクストラって先生が主張したプログラミングスタイルだよ。プログラミングパラダイムの先駆けってとこかな」
化太「難しいのぎゃ？」

機械「いや、全然簡単。基本三構造（順次／選択／反復）と、無条件ジャンプの禁止と、手続き化（入口／出口をそろえる）の三つだけ。最初の基本三構造でプログラムは殆ど書けるし、あとの二つは手続き化を覚えろってことだけだからね」

化太「もっと難しいのは要らんのぎゃ？」

機械「色々あるけど、最初は基本をきっちりマスターする方がいいと思う。手に馴染んでから、色々覚えたほうが、ありがたみもわかるしね」

化太「最初から便利なモノ覚えるべきぎゃ」

機械「不賛成。便利でも使いどころが少ないものは、後から覚えれば十分。最初に全て突っ込むと、

使いどころがわからなくなる。もっとも、あまり不自由な状態を続けるのは問題だけどね」

化太「こーぞーかだけでいいのぎゃ？」

機械「そうだな、シーケンスやコンテナ、それにポインタのこともある程度知っておいた方がいいかもしれない。データの表現方法とかは微妙だけどなぁ」

化太「二進法ぎゃ？」

機械「まぁね。二進法は必須じゃないけど覚えておいて損は無い。浮動小数点数の二進法表現とかは、基本からは要らんけどね。あと、算数の四則演算のほかに、集合論の基礎の基礎くらいは知っておくといい」

化太「しゅーごーろんぎゃ？」

機械「そう。ベン図とか、交わりとか結びとか、ある程度知っておくと、データ検索なんかがやりやすい」

化太「バブルソードはいいんぎゃ？」

機械「泡の剣って何だよ。バブルソートな。ああ、ソーティングアルゴリズムは初期のプログラミングの教材でよく使われたけど、今だと別にとりたててやる必要は無いなぁ」

化太「基本じゃないんきゃ？」

機械「別に。ソーティング用のライブラリなら、今ならどんな言語でも大体付いてるし。それより、プログラムで何ができるか、を知る方が必要だね。用意されたツールを使うための手続き／関数やメソッドの呼び出し方なんかは、勿論覚えておく必要があるけど」

化太「じゃ、ほとんど覚えること無いぎゃ」

機械「無いよ。実際、プログラミングの学習は、自分で考えて書くほうに重点を置くべきだと思う」

３．入門用最適な言語は？

化太「で、何を使うぎゃ？」

機械「プログラミング言語や環境か？　まぁ、俺の場合はPythonだけど、別に条件さえ揃えばなんでもいいよ」

化太「条件って何ぎゃ？」

機械「そーだな。まず、コンパイルしなくても使える環境が欲しいな。生のＣコンパイラやJavaコンパイラは、くり返しとなると負担になりそうだし」

化太「負担はいやぎゃ」

機械「あと、できるなら REPL 環境が使えるといいなぁ」

化太「れぷるぎゃ？」

機械「read-evaluate-print loop っていって、要するにシェルみたいなトコにコードを打ち込んでいくと、１行（論理行）づつ解釈してくれるインタープリタだよ。Lisp 系によく見られるけど、Python インタープリタとか Ruby の irb、あとちょっと感覚が違うけど、Smalltalk の workspace-transcript なんかもそうだな。ファイルを作らずにコードが実行できるから、素早く結果が見られて便利なんだ」

化太「よくわからんぎゃ」

機械「あとは、儀式が少ないほうがいいかな。Ｃが関数単位なのも Java がクラス単位なのも勿論必然的だし、それはそれで便利なんだけど、入門者には面倒なだけだから、そういうのが無いほうがいいかな」

化太「しんぷるいずべたーぎゃ」

機械「あと、Lisp や Force みたいに前置／後置記法の言語も、最初は避けたほうがいいかも。どちらも慣れれば便利なんだけど、最初は馴染みのある表現方法のほうがいいだろう」

化太「ぜんちこうちぎゃ？」

機械「前置記法は１ + １を+ １ １と書き、後置記法は１ １ +と書く書き方だよ」

化太「そんなんあるぎゃ？」

機械「あるよ」

化太「論外ぎゃ」

機械「いや、便利なんだよ、使えば。ただ、プログラミングを始めたばかりの人には負担なだけで」

化太「いーや、要らんぎゃ」

機械「まあいいや……あと、データが整数とか数値とかしか扱えない言語は止めた方がいいよなぁ。やっぱり文字列の処理も覚えておくべきだし」

化太「当然ぎゃ」

機械「欲を言うなら、文字列オブジェクトも使えて欲しいし、リストやハッシュみたいなモノも使えると便利だ」

化太「初めから要るぎゃ？」

機械「そうだね、便利だよ。それにプログラミングの考え方の幅も広がる」

化太「なら要るぎゃ」

機械「オブジェクト定義も、必須ではないけど欲しいな、できれば」

化太「入れるぎゃ」

機械「まぁ、この辺りの条件さえ整えばどんな言語でも入門できるよ。その言語特有の機能にあまり深く踏み込まない、という条件はつくけどね」

化太「深みにはまるとでてこれんぎゃ？」

機械「プログラミング入門っていう目的から外れるって意味。入門用の言語や環境はあくまで手段なんだから、入門書ではその言語や環境を使いこなすことを目的にしちゃマズい」

化太「言語特有の機能を説明しちゃいかんぎゃ？」

機械「全然説明しないってのは無理だろう。普通のプログラミングの実習に使う場合でも、他言語や環境では見られない機能とかある場合はあるから。ただ、最低限におさえておいたほうがいいってだけ」

化太「ミニマムアクセスだぎゃ」

機械「ただ、使用する言語に思い込みの激しい人ほど、言語の流儀をスタンダードだと無意識に主張してしまう傾向にあるから、そこらへんは意識的に注意する必要があるかもね」

４．サンプルコードは何を書く？

機械「最後は、サンプルコードに何を使うか決めれば終わりだな」

化太「早く決めるぎゃ」

機械「最初の hello word は一瞬で終わるから、次は電卓ソフトかな？」

化太「計算するぎゃ？」

機械「プログラムで計算するんじゃなくて、入力の結果を計算結果として返すプログラムを書くんだよ。入出力と evaluate 関数があれば、あとはループで簡単に書けちゃうからね」

化太「いばるえいとってなんぎゃ？」

機械「簡単に言えば、文字列を読み込んでプログラムとして実行する関数だな」

化太「ちっとも簡単じゃないぎゃ」

機械「まぁ、何度か使えばわかるよ。evaluate 関数が無いと、パーサを書かないといけないから、入門者用としてはちょっと面倒になるかもね」

化太「面倒は却下ぎゃ」

機械「ファイル入出力もある程度欲しいから、次はフィルタかな。テキストファイルを読み込んで HTML に加工するとか、簡単だし」

化太「簡単結構ぎゃ」

機械「ファイルの入出力ができれば、あとは簡易データベースとかがいいかも。ハッシュテーブルさえ使えれば、簡単なインデックス検索機能だけのデータベースならあっと言う間に書けるよ」

化太「あっという間ぎゃ」

機械「簡単なロジックゲームとかもいいかな。クラッシックな数当てゲームのＭＯＯとか」

化太「クラッシックぎゃ」

機械「ま、とりあえずそこまで書ければ、入門者は卒業」

化太「卒業ぎゃ」

機械「あとは、自分の目的に合った少し上級者向けの本を選べばいい」

化太「モテたいぎゃ」

機械「……モテねーよ、お前、絶対」

【おしまい】

バーチャル・バ○チャファイター2

written by Yさ

1. このゲームは
前作(THE100)に引き続き、貧相な画面のゲーム第三弾(^^;
コンピュータと対戦する、本格格闘アクションゲームです。

...って、うそです。ごめんなさい。
いわゆる格闘ゲーム風な攻防に数当て的要素をミックスしたものです。(. . . って、全然訳わかですね(^^;)
タイトルは某有名対戦格闘アクションゲームを彷彿させますが何の関係もありません。誤解です。

2. 動作環境は
例によって、最近のawkならOKだと思います。
ちなみに動作確認は、
GNU Awk 3.1.6, mawk 1.3.3 MBCS R27
で(WinXPのDOS窓で)行なってます。

3. 遊び方は
>gawk -f vvf2.awk[ENTER]
~~~~~~~~~~~~~~~~~~~~~~~~
等で起動するとゲームスタートです。
なお、ゲーム中に h か H または ? を入力すると、簡単なヘルプを表示します。
以下、その抜粋です。

1)5連続のCPU攻撃 g<oo>,c<hoki>,p<ar> を予測し、自分が勝つ
攻撃を指定します。(例 gcpGC[Enter] 大・小文字を区別)
2)勝敗判定結果に応じて負け側の体力`@`が減少します。
大文字と小文字が異なる攻撃の場合は負け側のダメージがよ
り大きくなります(カウンター)。どちらかの体力が0になるか
制限時間を過ぎるとラウンド終了です。
3)ラウンド終了時、残り体力が少ない方の生命力`*`が減少しま
す。どちらかの生命力が0になるとステージ終了です。
最終ステージまで勝ち抜いてください。

. . . 全然抜粋じゃないって(^^ゞ テヘ
ちなみに、簡単なヘルプは"HOW2PLAY.TXT"をそのまま表示しています。"HOW2PLAY.TXT"は60桁表示に収まる様に作っていますので、気に入らない場合などには自由に書き換えてくださいませ(^^;

CPU攻撃パターンはラウンド終了まで同じです。
ただし全て同じ攻撃(例 PPPPP)を指定していくつか外した場合は、CPU攻撃パターンが変わります。

最終ステージまで勝ち抜くか、途中で自分の全生命力が無くなると、ゲーム終了となります。
なお、スクリプトの LAST_STAGENO の値で最終ステージを変更できます。
数字は1～10を指定できます。デフォルトは5となっています。
※も少しヒントを出すと、
~~~~~~~~~~~~~~~~~~~~~~~~

```
split("COM,DOM,EOF,FOG,GOD,HAL,IBF,JFK,KGB,LUM", foe, ",");
```
の要素を増やせば(もしくは画面表示を気にしなければ)、10 より大きくできます d(^^;

4.開発環境など

ソースは、例によってかなり昔に、会社の行き帰りの満員電車で立ったまま作成したもの(C 言語)を、awk に移植したものです。

その当時は、前作(THE100)から丸２年ほど月日が流れていたのですが、相変わらずマシンは HP100LX でした v(^^;)

液晶パネル蓋の止め具はバカになっているし、電池入れ蓋はセロテープで止められていたりもするけれど、まだ元気よく動いていてくれていて、他のマシンには絶対到達できない便利さ＆タフさなのでした。しみじみ。

5. その他

本プログラムはフリーソフトです。

著作権は作者であるＹさにあります。

ただし転載、再配布、改造、消去は自由です。

(できましたら素晴らしい改造を加えた後に TSNet へ投稿してくださいませ)

また、このソフトを使用した事による損害が発生したとしても、損害に対しては一切の責任を負いかねます。

[スクリプト: vvf2.awk]

```
##  バーチャル・バ○チャファイター２  written by Ｙさ

BEGIN{
##
## 定数定義
##
LAST_STAGENO =5
# STAGE_LIMIT =10  ## 設定可能な最終ステージの限度
TIME_LIMIT =6
LIFE_LIMIT =2
STAT_LIMIT =60
BAR_LEN =5
BAR_UNIT =int(STAT_LIMIT/BAR_LEN)
DAMAGE =2
ATTACK_LEN =5

OFF =0
ON  =1

NO  =0
YES =1

YOU =0
```

```
COM =1
OTHER =2

# ステージ表示用状態定数
ROUND_START =10
ROUND_END   =11
STAGE_START = 0
STAGE_END   =21
GAME_END    =99
#  BOM -1- -2- -3- -4- -5- -6- -7- -8- -9- -10
split("COM,DOM,EOF,FOG,GOD,HAL,IBF,JFK,KGB,LUM", foe, ",");


# あたり判定用[自分,相手] : 1=自分が勝ち, -1=自分が負け, 0=互角
beatTbl["G","G"]=0;  beatTbl["G","C"]=1;  beatTbl["G","P"]=-1;
beatTbl["C","G"]=-1; beatTbl["C","C"]=0;  beatTbl["C","P"]=1;
beatTbl["P","G"]=1;  beatTbl["P","C"]=-1; beatTbl["P","P"]=0;


# 主な変数

# stage, round, times;
# stat[2];       ## 残り体力
# life[2];       ## 残り命
# hit[2];        ## あたり数
# attack[2];     ## 攻撃
# counterblow[2];
# thinkFlag;     ## CPU攻撃変更用フラグ
# thinkPutFlag;  ## CPU攻撃表示許可フラグ
# sc;            ## スコア

# bkupStat,bkupSC,bkupthinkFlag,bkupThinkPutFlag  ## 一時退避用

  # ゲームメイン
  doGame();
}


##
##  HELP 表示
##
function helpDisp(  msg){
  print "";
  while((getline msg < "how2play.txt")==1) printf("%s¥n", msg);
  close("how2play.txt");
  print "";
}
```

```
##
##  ステージ開始／終了表示
##
function putStage(sw ){  # <i>sw:表示フラグ
  if(sw==STAGE_START){ printf("¥n¥n<<< STAGE:%02d >>>¥n", stage); }
  if(sw==ROUND_START){ printf("¥n### ROUND:%02d ###     FIGHT!!!!!¥n", round); }
  if(sw==ROUND_END){         print "";
    if(times==0)             print  "     ++++++ TIME OVER ++++++";
         if(whichWin()==YOU) printf("     YOU WIN! (SCORE:%05d0)¥n",sc);
    else if(whichWin()==COM) printf("     YOU LOSE (SCORE:%05d0)¥n",sc);
    else                     printf("     ...DRAW  (SCORE:%05d0)¥n",sc);
  }
  if(sw==STAGE_END){
    if(whichWin()!=YOU)    printf("¥n     YOU WIN! (SCORE:%05d0)",sc);
    print "¥n     ...TRY NEXT STAGE";
  }
  if(sw==GAME_END){ print "¥n     ****** GAME OVER ******";
    if(life[YOU]>0){ printf("     Congratulations !!  (SCORE:%05d0)¥n",sc); }
    else           { printf("     ...YOU LOSE (STAGE:%02d SCORE:%05d0)¥n", stage, sc); }
  }
}

# 勝敗判定
function whichWin(){
       if(evalStat(stat[YOU])>evalStat(stat[COM])) return YOU;
  else if(evalStat(stat[YOU])<evalStat(stat[COM])) return COM;
  return OTHER;
}
function evalStat(stat){  # <i>stat:残体力
  if(stat<=0) return 0;
  return int(stat/BAR_UNIT)*10+((stat<BAR_UNIT)?(1):(0));
}

# 状況表示
function putStat(  bar, lbar, i){
  bar[YOU]=mkStatBarStr(stat[YOU]);  lbar[YOU]=mkStr(life[YOU], "*");
  bar[COM]=mkStatBarStr(stat[COM]);  lbar[COM]=mkStr(life[COM], "*");
  printf("%*s%s:%-*s [TIME:%02d] %*s:YOU%-*s¥n",
    LIFE_LIMIT,lbar[COM],foe[stage],BAR_LEN,bar[COM],
    times, BAR_LEN,bar[YOU],LIFE_LIMIT,lbar[YOU]);
}
function mkStr(len, ch,  dst){  # <i>len:文字列, ch:文字
  for(; len>0; --len) dst=dst ch;
  return dst;
}
function mkStatBarStr(stat){  # <i>stat:残体力
  if(0<stat && stat<BAR_UNIT) return ".";
```

```
  return mkStr(int(stat/BAR_UNIT), "@");
}

# あたり&ダメージ判定結果表示
function putJudge(){
  evalAttack();
  printf("   >> HIT %s=(%*s%d/%d%-*s)=YOU¥n",
    foe[stage],ATTACK_LEN,mkStr(counterblow[COM],"!"),hit[COM],
    hit[YOU],ATTACK_LEN,mkStr(counterblow[YOU],"!"));
}



##
##  あたり&ダメージ判定サブルーチン
##
function beatCom(yHand, cHand,  uY,uC){
  uY=toupper(yHand);  uC=toupper(cHand);
  return beatTbl[uY,uC]*(((((uY==yHand)?(1):(-1))*((uC==cHand)?(1):(-1)))<0)?(2):(1));
}
function evalAttack(  prevHand, allOneHand, i, eval, scTmp){
  hit[YOU]=hit[COM]=counterblow[YOU]=counterblow[COM]=0;
  allOneHand=ON;  # 仮
  prevHand=substr(attack[YOU],1,1);
  for(i=1; i<=ATTACK_LEN; ++i){
    if(substr(attack[COM],i,1)==" "){
      ++hit[YOU];  stat[COM]-=DAMAGE;
    }else if(substr(attack[YOU],i,1)==" "){
      ++hit[COM];  stat[YOU]-=DAMAGE;
    }else{
      eval=beatCom(substr(attack[YOU],i,1), substr(attack[COM],i,1));
      if(eval>0){
        ++hit[YOU]; stat[COM] -= (DAMAGE * eval); # YOU win
        if(eval>1) ++counterblow[YOU];
      }else if(eval<0){
        ++hit[COM]; stat[YOU] += (DAMAGE * eval); # COM win
        if(eval<-1) ++counterblow[COM];
      }
      scTmp+=eval;
    }
    if(prevHand!=substr(attack[YOU],i,1)) allOneHand=OFF;
    prevHand=substr(attack[YOU],i,1);
  }

  sc += (scTmp<0)?0:scTmp;

  # 同一繰り返しでオールヒットではない時は、CPU の攻撃を変更する
  if(allOneHand && 0<hit[YOU] && hit[YOU]<ATTACK_LEN){
    stat[YOU] -= DAMAGE*hit[COM];
```

```
    if(hit[COM]<=hit[YOU]){
      thinkFlag=thinkPutFlag=OFF;
    }
  }

  # オールヒット時は、以降の CPU の攻撃の表示を許可する
  if(hit[YOU]==ATTACK_LEN && (allOneHand || counterblow[YOU]==ATTACK_LEN))
    thinkPutFlag=ON;
}
function knockout(  dst){
  backupStat();
  evalAttack();  dst=(stat[COM]>0 && stat[YOU]>0)?NO:YES;
  restoreStat();
  return dst;
}
function backupStat(){
  bkupStat[YOU]=stat[YOU];
  bkupStat[COM]=stat[COM];
  bkupSC=sc;
  bkupThinkFlag=thinkFlag;
  bkupThinkPutFlag=thinkPutFlag;
}
function restoreStat(){
  stat[YOU]=bkupStat[YOU];
  stat[COM]=bkupStat[COM];
  sc=bkupSC;
  thinkFlag=bkupThinkFlag;
  thinkPutFlag=bkupThinkPutFlag;
}


##
##  人の攻撃
##
function inputManAttack(  i,len,tmp,p){
  printf("   >YOU: ");  getline tmp;
  # help ?
  p=toupper(substr(tmp, 1, 1)); if(p=="/")  p=toupper(substr(tmp, 2, 1));
  if(p=="H" || p=="?"){  #  help !
    helpDisp();
    putStat();          # 状況表示
    inputManAttack();  # 人の攻撃
    return;
  }
  # 入力した攻撃パターンの格納
  attack[YOU]="";
  for(len=i=0; i<length(tmp); ++i){
    p=toupper(substr(tmp, i+1, 1));
```

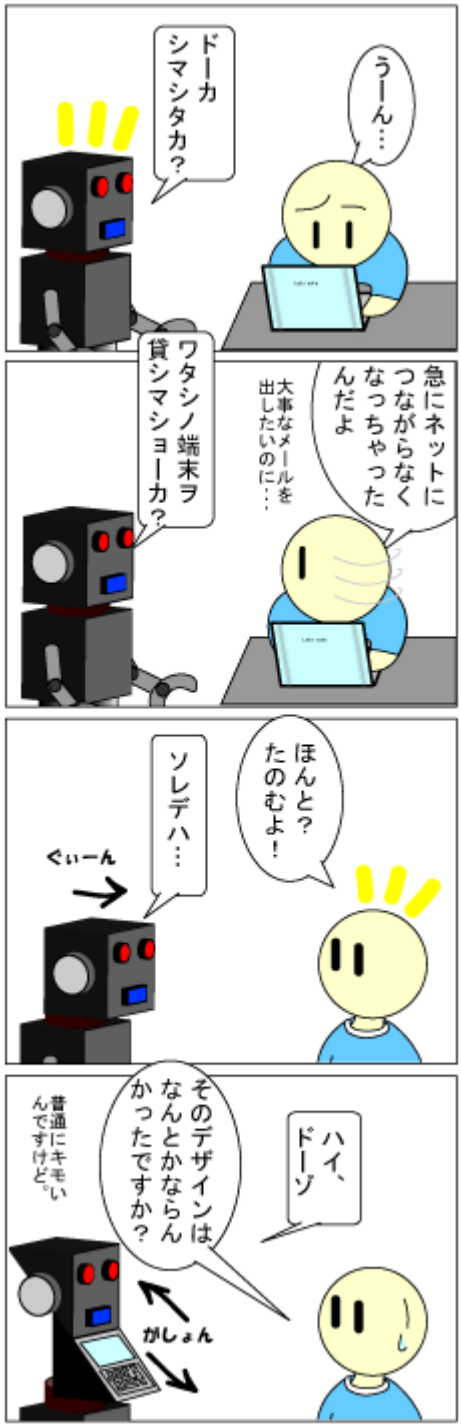
```
    if(p==" " || p=="C" || p=="G" || p=="P")
      attack[YOU] = attack[YOU] substr(tmp, i+1, 1);
    else
      attack[YOU] = attack[YOU] " ";
    ++len;
    if(len>ATTACK_LEN) break;
  }
  if(len<ATTACK_LEN) attack[YOU]=attack[YOU] mkStr(ATTACK_LEN-len, " ");
}


##
##  CPU 思考サブルーチン
##

# CPU の攻撃
function thinkComAttack(  i,sw,pat)
{
  if(thinkFlag) return;  # 攻撃を変更しない
  thinkFlag = ON;
  pat="gCpGcP";
  sw=0;
  if(stage<=2) sw=(rnd(3))?(int(rnd(40)/10)):((hit[YOU]==0)?3:2);
  if(hit[YOU]==0 || hit[COM]==5){
    if(stage>2) sw=(hit[COM]>0 && hit[COM]!=5)?3:2;
  }
  attack[COM]=mkStr(ATTACK_LEN, " "); # 仮
  for(; attack[COM]==mkStr(ATTACK_LEN, " "); sw=0){  # 全部空白の場合はやり直す
    if(sw==0 || sw==1){  # ランダム
      attack[COM]="";
      for(i=0; i<ATTACK_LEN; ++i) attack[COM]=attack[COM] substr(pat,rnd(6)+1,1);
    }else if(sw==2){     # 同一繰り返し
      attack[COM]=mkStr(ATTACK_LEN, substr(pat,rnd(6)+1,1));
    }else if(sw==3){     # 前と同じ
      # 何もしない
    }
  }
}
# CPU の攻撃表示
function putComAttack(sw,  tmp,i){  # <i>sw:表示/非表示フラグ
  if(sw) tmp=sprintf("%s", attack[COM]);
  else   tmp=mkStr(ATTACK_LEN, "?");
  printf("   >%s: %s",foe[stage],tmp);
}

function rnd(N){ return int(N * rand()); }  ##  乱数
```

```
##
##  ゲームメインループ
##
function doGame(){
  sc=0;
  srand();

  for(stage=1; stage<=LAST_STAGENO; ++stage){
    # ステージ開始
    life[YOU]=life[COM]=LIFE_LIMIT;
    putStage(STAGE_START);
    for(round=1; life[YOU]>0 && life[COM]>0; ++round){
      # ラウンド開始
      thinkFlag=thinkPutFlag=OFF;  # CPUの攻撃を変更する
      stat[YOU]=stat[COM]=STAT_LIMIT;
      putStage(ROUND_START);
      for(times=TIME_LIMIT; times>0 && stat[YOU]>0 && stat[COM]>0; --times){
        putStat();          # 状況表示
        inputManAttack();   # 人の攻撃入力
        thinkComAttack();   # CPUの攻撃決定
        if(times==1 || knockout()==YES) thinkPutFlag=ON;
        putComAttack(thinkPutFlag);
        putJudge();         # あたり&ダメージ判定結果表示

        # 体力がマイナスになったら0とする
        if(stat[YOU]<0) stat[YOU]=0;
        if(stat[COM]<0) stat[COM]=0;
      }
      # ラウンド終了
           if(whichWin()==COM){ --life[YOU]; }
      else if(whichWin()==YOU){ --life[COM]; }
      else                     { --life[YOU]; --life[COM]; }
      putStat();
      if(stat[YOU]>stat[COM]) sc+=(times*5+((stat[YOU]>0)?(int(stat[YOU]/10)):(0)))*10;
      putStage(ROUND_END);
    }
    # ステージ終了
    if(life[YOU]>0) sc+=stage*50;
    if(stage==LAST_STAGENO || life[YOU]<=0) break;
    putStage(STAGE_END);
  }
  # ゲーム終了
  putStage(GAME_END);
}
```

よしおさんとロボ太

海鳥作

「変形ロボ」

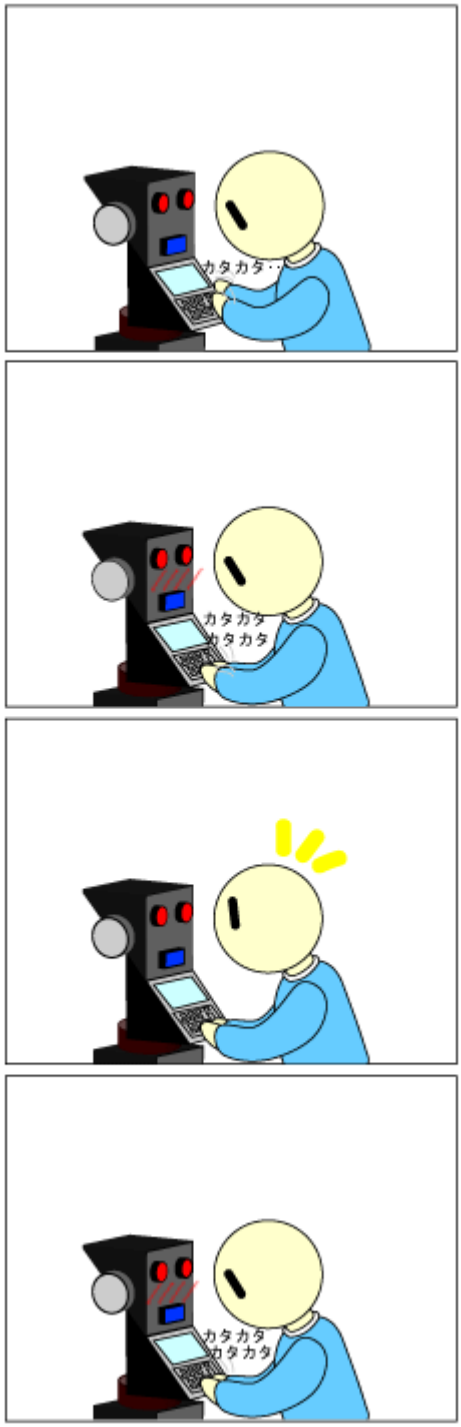


合体はしません(多分)。

よしおさんとロボ太

海鳥作

「続・変形ロボ」

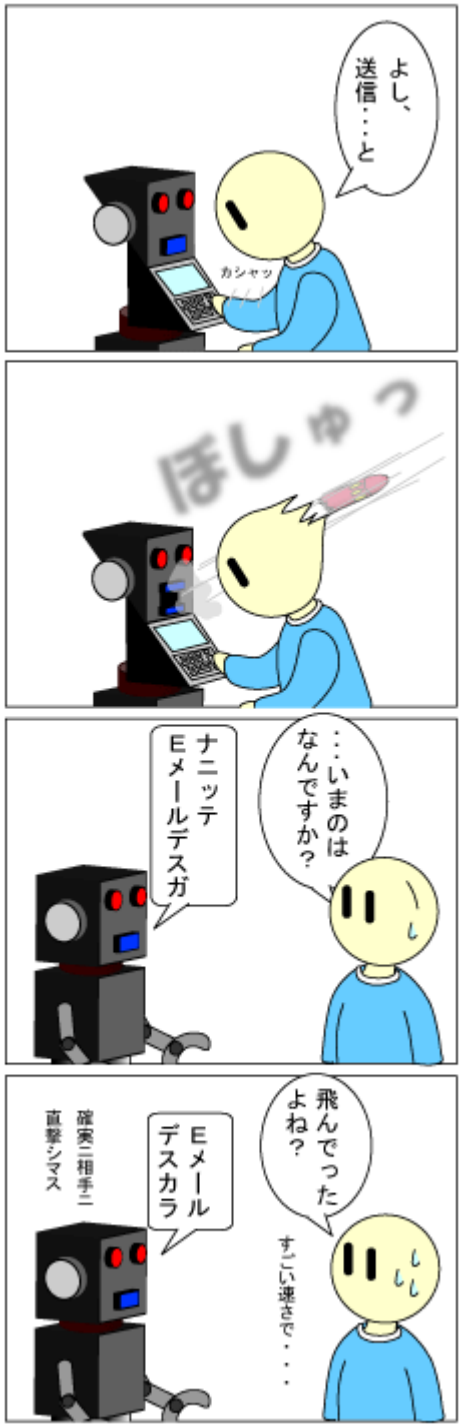


くすぐったいらしい。

よしおさんとロボ太

海鳥作

「続々・変形ロボ」



エマージェンシーメールの略？　（続く）

Wemo をつくる

jscripter

I. Wemo ってなに

Wemo は、「ウェモ」って読む。「ウィーモ」でもよいかもしれない。「Web Memo」、「Wiki Memo」、「We memo」の略称かもしれないし、単に Memo の最初の M を逆立ちさせただけかもしれない。

テキスト処理を信奉するテキスト主義者にとってはテキストエディタが唯一最強のツールだ。今や、Web はテキストと化しているので、テキスト主義者にとってはなんでもござれかもしれない。今をときめく HTML、XML、RDF などのマークアップ言語、Javascript、Perl、Ruby、Python、Tcl/Tk などのスクリプティング言語等々。世界がテキストになったようなものだ。テキストの有効性は非常に高まっている。そのような環境下でコンピュータを知的生産のために使うためには、まずメモに使いたいが、HTML の表現力を大いに活用することがポイントになるだろう。もうメモは単なるテキストではある必要はない。

著者は、「実践実用 Perl」(毎日コミュニケーションズ、2004 年)で、Web とデスクトップを融合するために、ローカル(デスクトップ)サーバーで動作する CGI(デスクトップ CGI と呼んでいる)の活用を提案した。デスクトップのフレームワークの中核に、「メモる」システムがあり、単なるメモだけでなく、カレンダーと連動させたり、ブックマークや RSS フィードやその記事などに関連付けて、メモを作成することができる。しかし、経験的にはメモはそれほど使わない結果に終わっている。Web 日記にメモ的なものを書き込んでしまうことが大きな原因だと考えている。ブログなどでは、時間さえあれば、メモを取るレベルよりはもう少し整理した形で、他の情報との関連付けも含めてまとめることができる。メモの役割は、断片的な情報を短時間に取りあえず定着することであり、メモを蓄積する場合には、メモを元にしたり、組み合わせることによって新しい情報を生み出すための仕組みを工夫する必要があるのだろう。

また、著者は、CodeZine(翔泳社)で、atom2msrdb.pl という rss2html.cgi の Perl5.8 版というべきものの開発を公開し、この中で、リレーショナルデータベースの MySQL をデスクトップ CGI のデータベースとして使う方法やデータ表示において Ajax を利用することも試みてきている。しかし、オブジェクト指向プログラミングに対する関心から、フレームワークという概念をもう一度検討してみたいという気持ちが強くなってきていて、全体を RSS も含めたシステムとして統合することは中断している。個々の CGI はデスクトップの中核で働いているのだが・・・

そこで、Wemo なんだが、「メモる」システムは jperl で書かれたが、Wemo は Perl5.8 で書いている。ただ、Wemo は「メモる」システムの単なる後継ではない。「メモる」システムではテキストの文字コードはシフト JIS であったが、Wemo の場合は UTF-8 である。そして、Wemo は Web アプリケーションとなるはずである。Wemo はローカルのサーバーではなく、Web で動く CGI である。

II. MVC フレームワーク

Web アプリケーションのフレームワークは、Ruby では Ruby on Rails、Perl では Catalyst が有名である。これらのフレームワークは、MVC、すなわち、Model-View-Controller のフレームワークと呼ばれている。

データをデータベースに格納して自在に取り扱おうと思えば、データ構造を決めなければならない。現実のデータをモデル化して取り扱う必要がある。そして、データは蓄積するだけでは役に立たない。

必要に応じて取り出し、表示する必要がある。すなわち、見せ方を決めなければならない。これが、View である。最後にデータを格納したり、取り出したり、表示したりする作業を制御しなくてはならない。Controller が必要なのである。

アプリケーションを作ろうと思えば、MVC の三つの役割をシステム的に考えていく必要がある。

III. CRUD の機能

CRUD は、Create-Retrieve-Update-Delete の略語である。Create は、データを作る機能を指す。Retrieve は、データを検索して取り出すこと、Update はデータを更新すること、Delete は不要なデータを削除することを指す。CRUD はデータを取り扱うための基本的なデータベースの制御機能(Controller)を意味し、それはその機能を実現するためのユーザーインターフェース(View)が必要なことに対応している。View と Controller はほぼ一体であると思われるが、Model は Controller を介して View と結び付けられると考えることもできる。CRUD はフレームワークの持つ機能を意味している。

IV. 自作のススメ

コンピュータを真の意味で役立てることは、自らプログラミングをして、自在にデータを加工し、新しい表現方法を獲得することである。市販のアプリケーションも素晴らしいがそれに満足していてはいけない。と無理やり動機付けをしているが、要はおもしろいから、興味があるからやるというのが本当のところだろう。そのようなことができるコンピューティング環境が正に現出したのである。ありとあらゆるフリーのプログラミング言語、テキスト化した Web 世界、低価格の PC とリーズナブルなインターネット利用コスト、これを活用しないということは考えられない。

本稿では、既に完全に出来上がったものを紹介するということではなくて、試行錯誤をしながら Wemo というアプリケーションを一緒に作っていこうという話題を提供する。おもしろいものができあがるかもしれないし、腰砕けに終わるかもしれないが、MVC フレームワークや CRUD の概念について、現実に即して考えることができるだろう。いずれにせよ、著者は何かを作り続けていくことは間違いない。

V. Wemo フレームワークの実際

CGI スクリプトの構成

ここで、Wemo に戻る。Wemo は今、産声を上げたばかりである。次の 4 つの CGI スクリプトからなる。

1. wemo_list.cgi　memo ファイルをリストアップして、日付とカテゴリとタイトルを表示する。
2. wemo.cgi　wemo 言語でマークアップされた memo を HTML に変換して表示する。
3. wemo_edit.cgi　memo ファイルを textarea に読み込み、編集可能とする。
4. wemo_save.cgi　編集後の memo ファイルを保存し、同時に HTML に変換して表示する。

「実践実用 Perl」の「メモる」システムを構成する memo1.cgi は wemo_save.cgi に移植し、memo1_edit.cgi は wemo_edit.cgi に移植した。wemo_save.cgi は wemo_edit.cgi から起動される。

「メモる」システムの memo.html(メモ編集を起動するインターフェース)に対応するものとして、単に memo ファイルをリストアップする wemo_list.cgi を作った。

Wemo システムと「メモる」システムの大きな違いは、Wemo は wemo 言語という独自のマークアップ言

語でメモを記録するのに対して、「メモる」システムはHTML でメモを保存することである。wemo.cgi は、単に wemo 言語で書かれたメモを HTML に変換して表示するだけの役目を果す。wemo.cgi は wemo_list.cgi から起動される。

セキュリティ

「メモる」システムはユーザーのデスクトップで動作させることが前提であるので、セキュリティを考えなくてよいが、Web アプリケーションである Wemo は、Wiki 同様に独自言語でテキストデータを保存する。これはデータを HTML 表示するときに使える機能を制限することによって、セキュリティを高めることを意図している。独自言語は、単にテキストデータの可読性を高めたり、入力の容易さを助けるだけのものではないのだろうと思う。

著者は今のところ、Web で動作する CGI の専門家ではないので、セキュリティ面についてはご注意願いたい。著者も注意を払って開発していくつもりだが、Web で利用する場合は自己責任でお願いしたい。著者の Web サイトで動作を試すことができるようにする予定である。

wemo 言語

wemo 言語は、「実践実用 Perl」の memol(memo language)言語や dml(diary markup language)言語とほぼ同じものである。また、CodeZine 記事の ddl(diary description language)言語[1]と同様のものである。ご参考にしていただきたい。「実践実用 Perl」(2004 年)は既に売り切れになっているので、CodeZine の関連記事を参照していただくのがよいだろう。

例えば、行頭に、「c:(半角スペース)テキストデータ」と置くと、テキストデータはカテゴリ(category)として読み取られる。 「t:(半角スペース)テキストデータ」ならタイトルということになる。意味は最終的には表現との対応関係があり、決められた HTML 表現に変換される。

意味を表す表記に加えて、単純に HTML に対応する表記もある。「p:(半角スペース)テキストデータ」とすれば、テキストデータは段落とみなされ、「 テキストデータ 」と変換される。

Wemo for PSP

Wemo はメモ作成ツールであり、Web アプリケーションなので、デスクトップよりはモバイル利用を想定したいと考えている。そのデバイスとして、PSP のインターネットブラウザの画面サイズを想定して View を考えている。使用デバイスによって CGI の動作をユーザーが制御するような仕組みを考えることができるだろう。

PSP のインターネットブラウザはメインメモリが 32MB[2]という制限からか、著者の Web 日記(更新日記)を表示しようとするとメモリ不足になって表示できない。表示データ量の上限があると思われる。今後の開発の過程で、様々な問題が出てくる可能性がある。デバイスにあわせた表示方法をコントロールする必要がある。

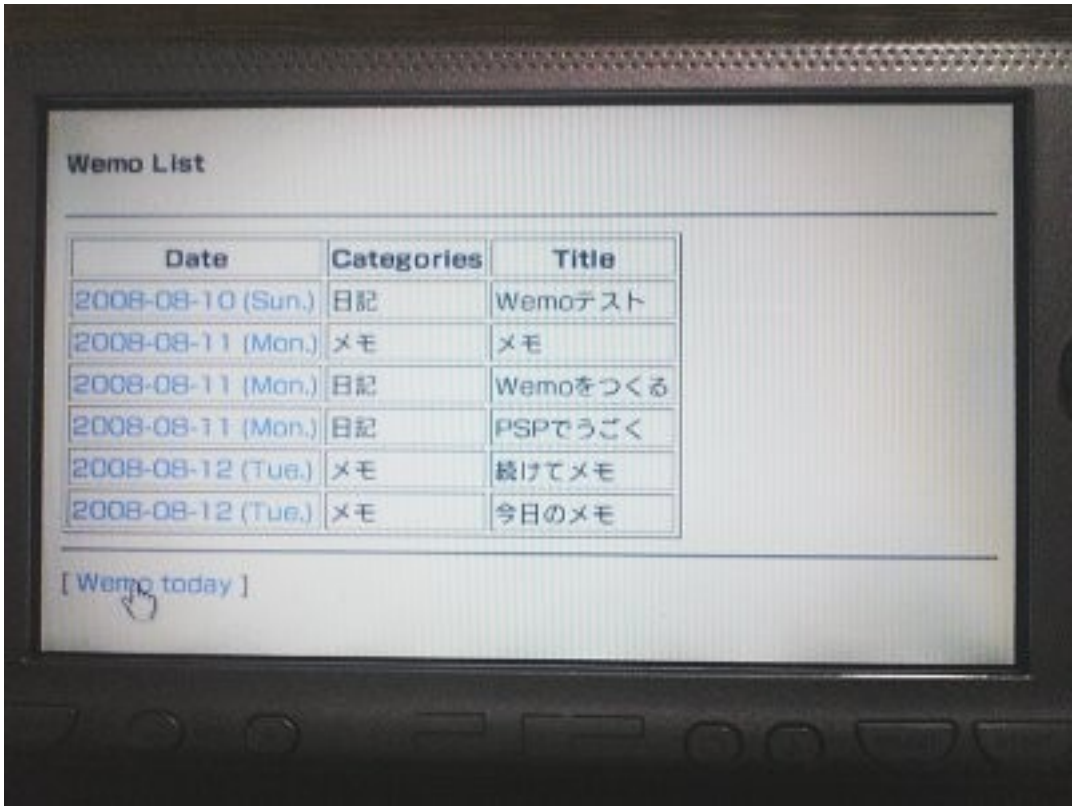
現段階での Wemo for PSP 動作画面を載せておく。

---

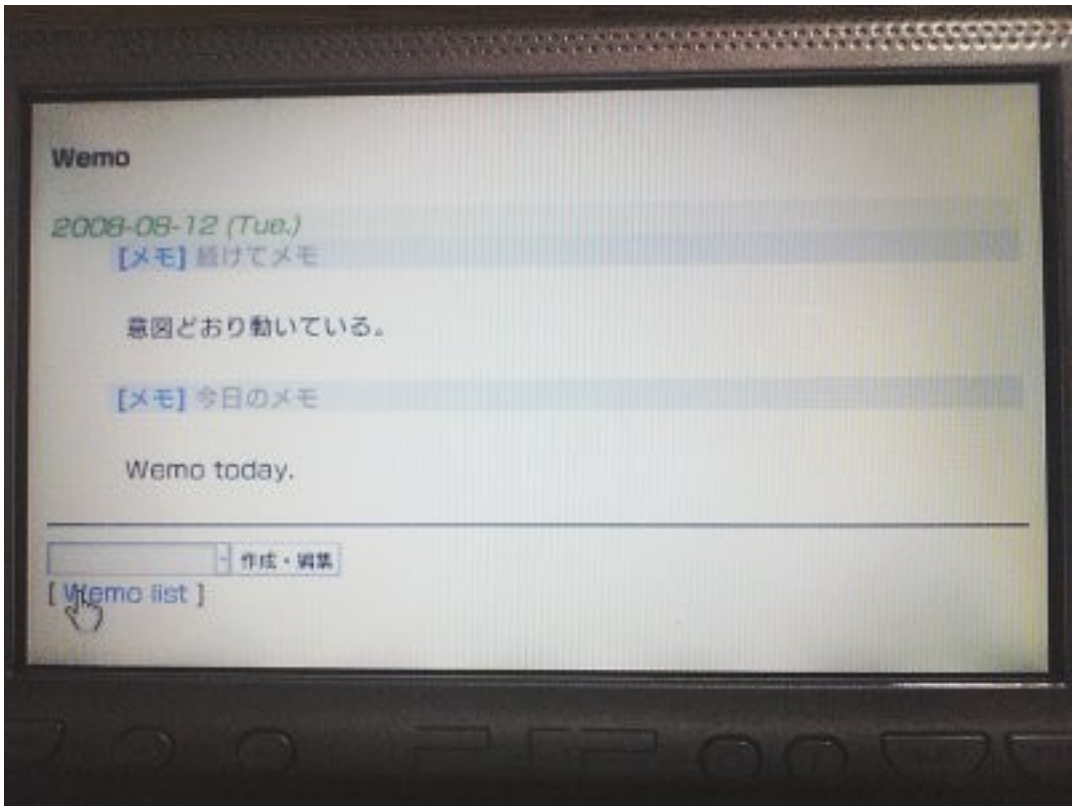
1 jscripter , 第 3 回 - 独自の日記記述言語から HTML 形式の Web 日記を出力・配信する
http://codezine.jp/a/article/aid/570.aspx
CodeZine, 翔泳社, 公開：06/10/12

2 著者の PSP は PSP-1000 であり、メインメモリは 32MB だが、PSP-2000 は 64MB 搭載している。

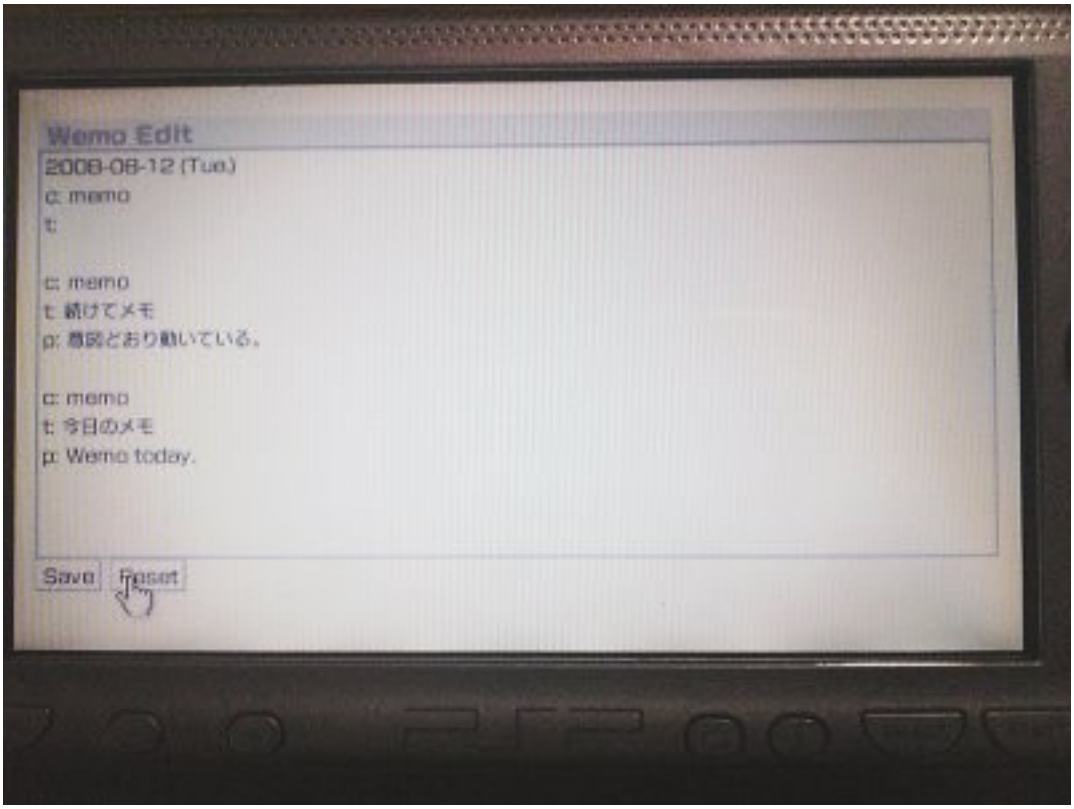
[wemo_list.cgi 実行画面]



[wemo.cgi/wemo_save.cgi 実行画面]



[wemo_edit.cgi 実行画面]



CGI スクリプトと今後の展開

最後に、Wemo システムの現在の CGI スクリプトを示しておく。基本的には「メモる」システムを Perl5.8 で動作するように移植し、さらに保存するデータは wemo 言語のままになるように改変している。strict や warnings プラグマを現段階では使用していない。「メモる」システムの他の機能に関わる狭雑物等が含まれたままであり、基本機能は動作するが、開発途中段階のコードであることをお断りしておく。今後の開発で、コードがどのように変化していくかを見ていきたい。

CGI のテストには、@nifty の LaCoocan サービスを利用している。Perl のバージョンは 5.8.7 であり、Time::Local などのモジュールが使用可能である。下記の URL でアクセスできる。

http://text.world.coocan.jp/cgi-bin/wemo_list.cgi

デスクトップ CGI のフレームワークにおいては CGI 一つ一つが一種のモジュールのようなものだと考えている。このこととオブジェクト指向や MVC フレームワークの概念がどのように関係していくのかということを本稿を通じて考えていきたい。

今後は、次のようなことを構想している。

1. データモデルの見直し
2. メモカレンダーの移植

3. 検索機能
4. 画像の添付表示
5. パスワード
6. 特定カテゴリ(形式)データのデータベース化機能(画像、青空文庫、千夜千冊、地理的情報、人名、Wikipedia)
7. Timeline表示
8. 使用可能デバイスの拡張(ケータイ等)
9. モジュール化
10. フレームワーク化
11. サービス化

すべての機能が、モバイルで使えるかどうかは疑問ではあるが。

疑問な点やアイデアがあれば、TSNETに書いていただいてもよいし、メールでもお問い合わせいただいたらよいと思う。歓迎である。

[wemo_list.cgi]

```
#!/usr/local/bin/perl

=head1 スクリプト名

wemo_list.cgi - Wemoデータリスト出力

=head1 概要

メモ格納ディレクトリの memo_yyyy-mm-dd.txt ファイルのリストとwemo.cgi起動用のファイルへのリンクを生成する。

=cut
use CGI qw(:cgi);

# 環境変数の取得
$docroot = "/homepage";# $ENV{'DOCROOT'};
$cgidir = "/cgi-bin";# $ENV{'CGIDIR'};

# メモを格納するディレクトリ名
$memodir = "memo";

# メモのカテゴリのASCII文字列をキーに日本語文字列を連想配列の値として取得
# メモ格納ディレクトリにある category.txt から読み込む
if(open(IN, "<$docroot/$memodir/category_utf-8.txt")){
        while(<IN>){
                chomp;
                ($catstr, $category) = split(/¥t/);
                $categories{$catstr} = $category;
        }
```

```
        close(IN);
}else{
        # category.txt が無ければ、次の連想配列のデータを使う
        %categories = (
                "favor" => "お気に入り",
                "diary" => "日記",
                "memo" => "メモ",
                "sched" => "スケジュール",
                "annot" => "註釈",
                "file" => "マイドキュメント",
        );
}

# メモの最初の部分の CGI 出力
print <<HEADER;
Content-type: text/html; charset=UTF-8

<HTML>
<HEAD>
<TITLE>Wemo List</TITLE>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="/mystyle.css">
</HEAD>
<BODY>
<H3>Wemo List</H3>
<hr>
<table border>
<tr><th>Date</th><th>Categories</th><th>Title</th></tr>
HEADER

opendir(DIR, "$docroot/$memodir");
@memo = grep(/^memo_/,readdir(DIR));
closedir(DIR);
foreach $memo (sort @memo){
        open(IN, "$docroot/$memodir/$memo");
        while(<IN>){
                chomp;
                if(/^¥d{4}-¥d{2}-¥d{2} ¥(¥w{3}¥.¥)$/){
                        $date = $_;
                }elsif(/^c: (.+)$/){
                        $category = $1;
                        $catsw = 1;
                }elsif(/^t: (.+)$/){
                        $title = $1;
                        $titsw = 1;
                }
                if($catsw == 1 && $titsw == 1){
                        print "<tr><td><a href=¥"$cgidir/wemo.cgi/$memo¥">$date</a></td>";
```

```
                        print "<td>$categories{$category}</td><td>$title</td></tr>¥n" ;
                        $catsw = 0;$titsw = 0;
                }
        }
        close(IN);
}
print "</table>¥n";
print "<hr>¥n";
print "[ <a href=¥"$cgidir/wemo_edit.cgi?today=yes¥">Wemo today</a> ]¥n";
print "</BODY></HTML>¥n";
```

[wemo.cgi]

```
#!/usr/local/bin/perl

=head1 スクリプト名

wemo.cgi - wemo 言語パーサー & メモ出力

=head1 概要

wemo_save.cgi が生成する wemo 言語で書いたメモテキストを HTML に変換し、CGI 出力を行う。

=cut
use CGI qw(:cgi);
use Time::Local;

# 環境変数の取得
$docroot = "/homepage";#$ENV{'DOCROOT'};
$cgidir = "/cgi-bin";#$ENV{'CGIDIR'};

# メモを格納するディレクトリ名
$memodir = "memo";

# メモのカテゴリの ASCII 文字列をキーに日本語文字列を連想配列の値として取得
# メモ格納ディレクトリにある category.txt から読み込む
if(open(IN, "<$docroot/$memodir/category_utf-8.txt")){
        while(<IN>){
                chomp;
                ($catstr, $category) = split(/¥t/);
                $categories{$catstr} = $category;
        }
        close(IN);
}else{
        # category.txt が無ければ、次の連想配列のデータを使う
        %categories = (
                "favor" => "お気に入り",
```

```
                "diary" => "日記",
                "memo" => "メモ",
                "sched" => "スケジュール",
                "annot" => "註釈",
                "file" => "マイドキュメント",
        );
}

# メモファイル名の取得
$infile = $ENV{'PATH_INFO'};

# メモファイル名からメモの日付を取得し、日付から曜日を timelocal 関数から取得
($date = $infile) =~ s/^¥/memo_(¥d{4}-¥d{2}-¥d{2})¥.txt$/$1/i;

# メモの最初の部分の CGI 出力
print <<HEADER;
Content-type: text/html; charset=UTF-8

<HTML>
<HEAD>
<TITLE>Wemo</TITLE>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="/mystyle.css">
</HEAD>
<BODY>
<H3>Wemo</H3>
<DL>
HEADER

#%articles = ();
# メモの入力ファイルをオープン
open(IN, "$docroot/$memodir$infile");

while(<IN>){
        chomp;
        $count++;
        # 日付行の次から処理を開始する
        if($_ =~ /^(¥d{4}-¥d{2}-¥d{2} ¥(¥w{3}¥.¥))$/i){
                $insw = 1;# フラグを立てる
                print "<DT>$1<DD>¥n";
                next;
        }
        if($insw == 1){
                if($_ =~ /<.+>/ && $pre != 1){
                        next;
                }
                # カテゴリデータの処理
                if($_ =~ /^C:¥s+(.*)$/i){
```

```
		$category = $1;# カテゴリデータを取得する
		next unless $category;# カテゴリデータがない場合はスキップ
		# カテゴリとタイトルデータが揃ったら処理をする
		# タイトルデータが空の場合はスキップ
		if($title ne ""){
			$catid = time + $catcount;
			$articles{"${category}_$catid"} = $title;
			# CGI 出力
			print "<div class=¥"emph¥">¥n";
			print "<A HREF=¥"/$memodir/memo_index.html#$category¥">";
			print "[$categories{$category}]</A> ";
			print "<A NAME=¥"${category}_$catid¥">$title</A></div>¥n";
			$title = "";
			$category = "";
			$catcount++;
		}
	# タイトルデータの処理
	}elsif($_ =~ /^T:¥s+(.*)$/i){
		$title = $1;# タイトルデータを取得する
		next unless $title;# タイトルデータがない場合はスキップ
		# タイトルにハイパーリンクを張る場合の処理
		$title =~ s/¥[U:¥s+([^;]+);T:¥s+([^¥]]+)¥]/<a href="$1">$2<¥/a>/i;
		$title =~ s/¥[U:¥s+([^;¥]]+)¥]/<a href="$1">$1<¥/a>/i;
		# カテゴリとタイトルデータが揃ったら処理をする
		# カテゴリデータが空の場合はスキップ
		if($category ne ""){
			$catid = time + $catcount;
			$articles{"${category}_$catid"} = $title;
			# CGI 出力
			print "<div class=¥"emph¥">¥n";
			print "<A HREF=¥"/$memodir/memo_index.html#$category¥">";
			print "[$categories{$category}]</A> ";
			print "<A NAME=¥"${category}_$catid¥">$title</A></div>¥n";
			$category = "";
			$title = "";
			$catcount++;
		}
	# パラグラフ（段落）データの処理
	}elsif($_ =~ /^(?:DE:|P:)¥s+(.*)$/i){
		$_ = $1;
		next unless $_;# データが空の場合はスキップ
		&linking($_);
		print "<p>$_</p>¥n";# CGI 出力
		next;
	# 整形済みテキストデータの処理 (1)
	}elsif($_ =~ /^PRE:$/i){
		print "<pre>¥n";# CGI 出力
		$pre = 1;# フラグを立てる
```

```
        next;
# 整形済みテキストデータの処理 (2)
}elsif($_ =~ /^¥/PRE:/i){
        print "</pre><br>¥n";# CGI 出力
        $pre = 0;# フラグを元に戻す
        next;
# 番号なしリストデータの処理 (1)
}elsif($_ =~ /^UL:$/i){
        print "<ul>¥n";# CGI 出力
        $ul = 1;# フラグを立てる
        next;
# 番号なしリストデータの処理 (2)
}elsif($_ =~ /^¥/UL:$/i){
        print "</ul><br>¥n";# CGI 出力
        $ul = 0;# フラグを元に戻す
        next;
# 番号付きリストデータの処理 (1)
}elsif($_ =~ /^OL:$/i){
        print "<ol>¥n";# CGI 出力
        $ol = 1;# フラグを立てる
        next;
# 番号付きリストデータの処理 (2)
}elsif($_ =~ /^¥/OL/i){
        print "</ol><br>¥n";# CGI 出力
        $ol = 0;# フラグを元に戻す
        next;
# 階層型リストデータの処理 (1)
}elsif($_ =~ /^ULS:$/i){
        $blevel = 0;
        print "<ul>¥n";# CGI 出力
        $uls = 1;# フラグを立てる
        next;
# 階層型リストデータの処理 (2)
}elsif($_ =~ /^¥/ULS:$/i){
        while($blevel > 0){
                print "¥t" x $blevel;# CGI 出力
                print "</ul>¥n";
                $blevel--;
        }
        print "</ul><br>¥n";# CGI 出力
        $uls = 0;# フラグを元に戻す
        next;
# 定義リストデータの処理 (1)
}elsif($_ =~ /^DL:$/i){
        # <dl>タグの前にタブを入れて、メモ開始の<dl>タグと区別
        print "¥t<dl>¥n";# CGI 出力
        $dl = 1;# フラグを立てる
        next;
```

```
		# 定義リストデータの処理 (2)
		}elsif($_ =~ /^¥/DL/i){
			# </dl>タグの前にタブを入れて、メモ終了の</dl>タグと区別
			print "¥t</dl><br>¥n";# CGI 出力
			$dl = 0;# フラグを元に戻す
			next;
		# 表データの処理 (1)
		}elsif($_ =~ /^ta:$/i){
			$ta = 1;# フラグを立てる
			$count = 0;
			next;
		# 表データの処理 (2)
		}elsif($_ =~ /^¥/ta:$/i){
			print "</TABLE><BR>¥n";# CGI 出力
			$ta = 0;# フラグを元に戻す
			next;
		}elsif($_ =~ /^bq:$/i && $pre != 1){
			$bq = 1;
			print "<BLOCKQUOTE>¥n";
			next;
		}elsif($_ =~ /^¥/bq:$/i && $pre != 1){
			$bq = 0;
			print "</BLOCKQUOTE>¥n";
			next;
		# 改行データの処理 (1)
		}elsif($_ =~ /^br:$/i){
			$br = 1;# フラグを立てる
			next;
		# 改行データの処理 (2)
		}elsif($_ =~ /^¥/br:$/i){
			$br = 0;# フラグを元に戻す
			print "<br>¥n";# CGI 出力
			next;
		# memol 言語で記述した画像データの処理
		}elsif($_ =~ /^¥[[^¥]]+¥]$/ && $pre != 1 && $ul != 1 && $ol != 1 && $uls !=
1 && $dl != 1 && $ta != 1 && $br != 1 && $bq != 1){
			&linking($_);
			print $_, "¥n";# CGI 出力
		# その他の行の処理
		}elsif($_ !~ /^¥w{1,3}:$/ && $line !~ /^¥w{1,3}: .+$/ && $pre != 1 && $ul !
=1 && $ol != 1 && $uls != 1 && $dl != 1 && $ta != 1 && $br != 1 && $bq != 1){
			&linking($_);
			print $_, "¥n";# CGI 出力
		}
		# フラグが立っている状態でのデータ処理
		# 整形済みテキストデータの処理 (3)
		# <>記号のみ実体参照エンコードする
		if($bq == 1){
```

```
        &linking($_);
        print "$_¥n";
}
if($pre == 1){
        $line = &entities_encode($_);
        print $_, "¥n";# CGI 出力
}
# 番号なしリストデータの処理 (3)
if($ul == 1){
        &linking($_);
        print "<li>$_¥n";# CGI 出力
}
# 番号付きリストデータの処理 (3)
if($ol == 1){
        &linking($_);
        print "<li>$_¥n";# CGI 出力
}
# 階層型リストデータの処理 (3)
if($uls == 1){
        &linking($_);
        if($line =~ /^([ ¥t]*)[^ ¥t].*$/){
                $level = length $1;
                if($blevel > $level){
                        $diff = $blevel - $level;
                        if($diff >= 1){
                                for($i = 1;$i<=$diff;$i++){
                                        $ts = "¥t" x ($blevel - $i + 1);
                                        print "$ts</ul>¥n";# CGI 出力
                                }
                        }
                        $_ =~ s/^([ ¥t]*)([^ ¥t].*)$/$1<li>$2/;
                }elsif($blevel == $level){
                        $_ =~ s/^([ ¥t]*)([^ ¥t].*)$/$1<li>$2/;
                }elsif($blevel < $level){
                        $diff = $level - $blevel;
                        if($diff >= 1){
                                for($i = 1;$i<=$diff;$i++){
                                        $ts = "¥t" x ($blevel + $i);
                                        print "$ts<ul>¥n";# CGI 出力
                                }
                        }
                        $_ =~ s/^([ ¥t]*)([^ ¥t].*)$/$1<li>$2/;
                }
        }
        $blevel = $level;
        print $_,"¥n";# CGI 出力
}
# 定義リストデータの処理 (3)
```

```
                if($dl == 1){
                        if($_ =~ /^([^:]+:) (.*)$/){
                                $prop = $1;
                                $str = &linking($2);
                                print "¥t¥t<dt>$prop¥n¥t¥t<dd>$str¥n";# CGI 出力
                        }
                }
                # 改行データの処理 (3)
                if($br == 1){
                        &linking($_);
                        $_ =~ s/^(.*)$/$1<br>/;
                        print $_,"¥n";# CGI 出力
                }
                # 表データの処理 (3)
                if($ta == 1){
                        $count++;
                        if($count == 1){
                                $caption = &linking($_);
                                # CGI 出力
                                print "<TABLE BORDER=7>¥n";
                                print "<CAPTION ALIGN=top><B>$caption</CAPTION>¥n";
                        }elsif($count == 2){
                                # 表の項目データを取得して、HTML を出力します。
                                &linking($_);
                                @heads = split(/:/, $_);
                                print "<TH>";# CGI 出力
                                for($i=0; $i<$#heads; $i++){
                                        print "$heads[$i]</TH><TH>";# CGI 出力
                                }
                                print "$heads[$#heads]</TH>¥n";# CGI 出力
                        }else{
                                # 規則的なテキストデータから１行ずつ変換して出力します。
                                &linking($_);
                                @fd = split(/:/, $_);
                                print "<TR>¥n<TD>";# CGI 出力
                                for($i=0; $i<$#fd; $i++){
                                        print "$fd[$i]</TD><TD>";# CGI 出力
                                }
                                print "$fd[$#fd]</TD>¥n</TR>¥n";# CGI 出力
                        }
                }
        }
}
close(IN);

# メモ HTML 最終部分の CGI 出力
print "</DL>¥n<HR>¥n";
```

```
# メモ作成・編集用フォームの CGI 出力
print "<FORM action=¥"$cgidir/wemo_edit.cgi¥">¥n";
# <<重要>> INPUT タグの hidden 属性で、メモの日付 date を $date として
# wemo_edit.cgi に渡すことによって、どのメモファイルを処理するのかが同定される
print "<INPUT TYPE=¥"hidden¥" NAME=¥"date¥" VALUE=¥"$date¥">¥n";
# SELECT タグによるカテゴリの選択メニューを出力する
print "<SELECT name=¥"category¥" size=1>¥n";
print "<OPTION selected></OPTION>¥n";

# OPTION タグで、カテゴリの選択メニューを作成する
foreach $catstr (sort keys(%categories)){
        print "<OPTION value=¥"$catstr¥">$categories{$catstr}</OPTION>¥n";
}
# フォームの完成
print "</SELECT><INPUT TYPE=¥"submit¥" VALUE=¥"作成・編集¥"></FORM>¥n";
print "[ <a href=¥"$cgidir/wemo_list.cgi¥">Wemo list</a> ]¥n";

# CGI 出力の終了
print "</BODY>¥n</HTML>¥n";

# 文字データ中の memo1 言語文字列の処理
sub linking{
        $catid = time + $catcount;# アンカー名を確実に区別して生成する

        # ハイパーリンクの処理
        # [u: URL;T: タイトル]
        $_[0] =~ s/¥[U:¥s+([^;]+);T:¥s+([^¥]]+)¥]/<a href="$1">$2<¥/a>/gi;
        # [u: アンカーベース名;t タイトル]
        $_[0] =~ s/¥[N:¥s+([^;]+);T:¥s+([^¥]]+)¥]/<a name="${1}_$catid">$2<¥/a>/i;
        # [u: URL]
        $_[0] =~ s/¥[U:¥s+([^;¥]]+)¥]/<a href="$1">$1<¥/a>/gi;

        # 画像データの処理
        # [i: 画像 URL]
        $_[0] =~ s/¥[I:¥s+([^;¥]]+)¥]/<IMG SRC="$1" ALIGN=LEFT HSPACE=20 ALT="$1">/gi;
        # [i: 画像 URL;t: タイトル]
        $_[0] =~ s/^¥[I:¥s+([^;]+);T:¥s+([^¥]]+)¥]$/<IMG SRC="$1" ALIGN=LEFT HSPACE=20
ALT="$1">$2<BR CLEAR=ALL>/i;
        # [i: 画像 URL;u: 画像にリンクする URL;t: タイトル]
        $_[0] =~ s/^¥[I:¥s+([^;]+);U:¥s+([^;]+);T:¥s+([^¥]]+)¥]$/<IMG SRC="$1" ALIGN=LEFT
HSPACE=20 ALT="$1"><a href="$2">$3<¥/a><BR CLEAR=ALL>/i;
        # [u: 画像にリンクする URL;i: 画像 URL;t: タイトル]
        $_[0] =~ s/^¥[U:¥s+([^;]+);I:¥s+([^;]+);T:¥s+([^¥]]+)¥]$/<a href="$1"><IMG SRC="$2"
ALIGN=LEFT HSPACE=20 ALT="$2"><¥/a>$3<BR CLEAR=ALL>/i;
        # [u: 画像にリンクする URL;i: 画像 URL;u: タイトルにリンクする URL;t: タイトル]
        $_[0] =~ s/^¥[U:¥s+([^;]+);I:¥s+([^;]+);U:¥s+([^;]+);T:¥s+([^¥]]+)¥]$/<a
href="$1"><IMG SRC="$2" ALIGN=LEFT HSPACE=20 ALT="$2"><¥/a><a href="$3">$4<¥/a><BR
CLEAR=ALL>/i;
```

```
        # 『』で囲まれた部分を引用部として処理する
        $_[0] =~ s/『([^』]+)』/『<cite>$1<¥/cite>』/ig;

        # URL 文字列に URL 文字列の表すハイパーリンクを張る
        # URL 文字列の次の文字が URL 文字列を構成する文字列以外であり、
        # かつマークアップ記号ではない場合、
        $_[0] =~ s/(http:¥/¥/[#%a-z0-9¥-_.!~*'¥(¥);¥/?:@&=+¥$,]+)([^<>"#%a-z0-9¥-_.!~*'¥
(¥);¥/?:@&=+¥$,]+)/<a href="$1">$1<¥/a>$2/ig;
        # URL 文字列の並びの次が行末である場合、
        $_[0] =~ s/(http:¥/¥/[#%a-z0-9¥-_.!~*'¥(¥);¥/¥?:@&=+¥$,]+)$/<a href="$1">$1<¥/a>/i;

        $catcount++;
        return $_[0];
}

# 現在の日付を yyyy-mm-dd 形式で得る
sub date{
        ($mon = (localtime)[4] + 1) =~ s/^(¥d)$/0$1/;
        ($mday = (localtime)[3]) =~ s/^(¥d)$/0$1/;
        $year = (localtime)[5] + 1900;
        return $year . "-" . $mon . "-" . $mday;
}

# <>実体参照エンコード（<pre>タグの中で使う）
sub entities_encode{
  my($str) = @_;
  $str =~ s/</&lt;/g;
  $str =~ s/>/&gt;/g;
  return $str;
}

# URL エンコーディング
sub jurl_encode{
  my($str) = @_;
  $str =~ s/([^a-z0-9¥-_.!*'¥(¥)~ ])/sprintf("%%%02X", ord($1))/egi;
  $str =~ s/ /+/g;
  return $str;
}
```

[wemo_edit.cgi]

```
#!/usr/local/bin/perl

=head1 スクリプト名

wemo_edit.cgi - メモ作成編集（Wemo システムの CGI の一つ）
```

=head1 概要

HTML フォーム等からの変数入力に基づいて、wemo 言語による HTML メモのテキストフォームを生成する。保存時に、wemo_save.cgi を起動し、メモテキストを生成すると同時にHTMLに変換して表示する。

=cut
```
use CGI qw(:cgi);

# 環境変数の取得とメモを格納するディレクトリ名の設定
$docroot = "/homepage";#$ENV{'DOCROOT'};# HTTP サーバーのドキュメント格納ディレクトリ
$cgidir = "/cgi-bin";#$ENV{'CGIDIR'};# HTTP の CGI を格納する相対パス
$memodir = "memo";# メモ格納ディレクトリ名

# メモ入力 HTML 先頭部分 CGI 出力
print <<TOP;
Content-type: text/html; charset=UTF-8

<HTML>
<HEAD>
<TITLE>Wemo</TITLE>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="/mystyle.css">
</HEAD>
<BODY>
<div class="emph">Wemo Edit</div>
TOP

# フォームのデータに基づいてメモファイル名を設定する
# param('today') の値が yes であれば今日の日付のメモを作成する
# param('date') の値があれば、その値の日付のメモを作成する
# いずれでもなければ、現在の日付のメモを指定する
if(param('today') eq 'yes'){
        $date = &date;
}elsif(param('selecteddate')){
        $date = param('selecteddate');
}elsif(param('date')){
        $date = param('date');
}elsif($ENV{'PATH_INFO'}){
        ($date = $ENV{'PATH_INFO'}) =~ s/^¥/memo_(¥d{4}-¥d{2}-¥d{2})¥.txt$/$1/i;
}
if($date !~ /^¥d{4}-¥d{2}-¥d{2}$/){
        print "Wemo's date: ${date} is invalid!";
        exit;
}
$infile = "memo_${date}.txt";# メモファイル名の設定
```

```
# メモ入力フォーム先頭部分の CGI 出力
# メモファイル名 $infile は CGI のパス名の後に / で区切って渡される
# wemo.cgi は環境変数 PATH_INFO から /$infile として受け取る
print <<HEADER;
<form method="POST" action="$cgidir/wemo_save.cgi/$infile" target="main">
<textarea rows=15 cols=60 name="memo">
HEADER

# メモファイルがあれば読み込み、メモの日付行に続いて、wemo 言語を出力する
if(open(IN, "<$docroot/$memodir/$infile")){
        while(<IN>){
                print;
                if(/^(¥d{4}-¥d{2}-¥d{2}) ¥(¥w{3}¥.¥)$/i){
                        $date = $1;
                        &printwemo;
                }
        }
        close(IN);
# メモファイルがなければ、wemo 言語で日付行等を出力する
}else{
        print "da:¥n";
        &printwemo;
}

# メモ入力フォーム最終部分を CGI 出力する
print <<FOOTER;
</textarea>
<br>
<input type="submit" value="Save">
<input type="reset" value="Reset">
</form>
</BODY>
</HTML>
FOOTER

sub printwemo{
        # memo.html の「今日」のラジオボタンからの入力があれば、
        if(param('today')){
                print "c: memo¥n";
                print "t: ¥n¥n";
        }elsif(param('rndate')){
                print "c: memo¥n";
                print "t: ",param('rndate'),": ",param('title'),"¥n";
                print "p: [u: ",param('rnurl'),";t: ";
                print param('rndate'),": ",param('title'),"]¥n";
        # memo.html の「指定する日」のテキストフォームからの入力があれば、
        }elsif(param('selecteddate')){
                print "c: memo¥n";
```

```
                print "t: ¥n¥n";
        # 「お気に入り」検索結果から起動する場合、
        # date、url、title のデータを受け取る
        }elsif(param('url')){
                print "c: favor¥n";
                print "t: ",param('title'),"¥n";
                print "p: [u: ",param('url'),"]¥n¥n";
        # 「マイドキュメント」検索結果から起動する場合、
        # date、docurl、title のデータを受け取る
        }elsif(param('docurl')){
                print "c: file¥n";
                print "t: ",param('title'),"¥n";
                print "p: [u: ",param('docurl'),"]¥n¥n";
        # 「註釈を書く」のリンクから起動する場合、
        # annot、title、name のデータを受け取る
        }elsif(param('annot')){
                print "c: annot¥n";
                print "t: ",param('annot'),": 「",param('title'),"」のメモへの註釈¥n";
                print "p: [u: /$cgidir/wemo.cgi/memo_",param('annot'),".txt#";
                print param('name'),"]¥n¥n";
        # メモカレンダーから起動する場合、
        # date、sched のデータを受け取る
        }elsif(param('sched')){
                print "c: sched¥n";
                print "t: ¥n";
                print "p: ¥n¥n";
        # メモ作成・編集 HTML のメニュー選択フォームから起動する場合、
        # category のデータを受け取る
        }elsif(param('category')){
                print "c: ",param('category'),"¥n";
                print "t: ¥n";
                print "p: ¥n¥n";
        }
}

# yyyy-mm-dd 形式で現在の日付を返す
sub date{
        my($mday,$mon,$year) = (localtime)[3..5];
        $mon += 1;
        $year += 1900;
        return sprintf("%4d-%1.2d-%1.2d", $year, $mon, $mday);
}
```

[wemo_save.cgi]

```
#!/usr/local/bin/perl
```

=head1 スクリプト名

wemo.cgi - Wemo 言語パーサー & メモ出力

=head1 概要

wemo_edit.cgi が生成するメモ入力フォームに wemo 言語で書いたメモを HTML に変換し、CGI 出力とメモ格納ディレクトリへの memo_yyyy-mm-dd.txt ファイルの出力を行う。

```
=cut
use CGI qw(:cgi);
use Time::Local;

# 環境変数の取得
$docroot = "/homepage";#$ENV{'DOCROOT'};
$cgidir = "/cgi-bin";#$ENV{'CGIDIR'};

# メモを格納するディレクトリ名
$memodir = "memo";
# メモを格納するディレクトリが存在しなければ作成する
unless(-e "$docroot/$memodir"){
        mkdir "$docroot/$memodir", 0666;
}

# メモファイル名の取得
$outfile = $ENV{'PATH_INFO'};

# メモファイル名からメモの日付を取得し、日付から曜日を timelocal 関数から取得
($date = $outfile) =~ s/^¥/memo_(¥d{4}-¥d{2}-¥d{2})¥.txt$/$1/i;
($year,$mon,$mday) = split(/-/,$date);
$mon -= 1;# 月データを timelocal 関数入力用に調整
$time = timelocal(0,0,0,$mday,$mon,$year);
$youbi = ('Sun','Mon','Tue','Wed','Thu','Fri','Sat')[(localtime($time))[6]];
$mon += 1;# 月数データに戻す

# メモのカテゴリの ASCII 文字列をキーに日本語文字列を連想配列の値として取得
# メモ格納ディレクトリにある category_utf-8.txt から読み込む
if(open(IN, "<$docroot/$memodir/category_utf-8.txt")){
        while(<IN>){
                chomp;
                ($catstr, $category) = split(/¥t/);
                $categories{$catstr} = $category;
        }
        close(IN);
}else{
        # category.txt が無ければ、次の連想配列のデータを使う
        %categories = (
                "favor" => "お気に入り",
```

```
                "diary" => "日記",
                "memo" => "メモ",
                "sched" => "スケジュール",
                "annot" => "註釈",
                "file" => "マイドキュメント",
        );
}

# メモの出力ファイルをオープン
open(OUT, ">$docroot/$memodir$outfile");

# メモの最初の部分の CGI 出力
print <<HEADER;
Content-type: text/html; charset=UTF-8

<HTML>
<HEAD>
<TITLE>Wemo</TITLE>
<META HTTP-EQUIV="Content-Type" content="text/html; charset=UTF-8">
<link rel="stylesheet" type="text/css" href="/mystyle.css">
</HEAD>
<BODY>
<H3>Wemo</H3>
<DL>
HEADER

# CGI で受け取るメモデータ param('memo')を改行で行単位に分割して処理する
@memo = split(/¥r¥n/, param('memo'));# Windows の改行 ¥r¥n の並びで分割する
foreach $line (@memo){
        $linecount++;
        # メモを最初に作成する場合、最初の行が wemo 言語の始まりにマッチする
        if($linecount == 1 && $line =~ /^da:/i){
                print "<DT>${date} (${youbi}.)<DD>¥n";
                print OUT "${date} (${youbi}.)¥n";
                $insw = 1;# 処理を開始するフラグ（旗）を立てる
                next;
        }
        # メモが既にある場合、日付行の次から処理を開始する
        if($line =~ /^(¥d{4}-¥d{2}-¥d{2} ¥(¥w{3}¥.¥))$/i){
                $insw = 1;# フラグを立てる
                print "<DT>$1<DD>¥n";
                print OUT $1,"¥n";
                next;
        }
        if($insw == 1){
                if($line =~ /<.+>/ && $pre != 1){
                        next;
                }else{
```

```
        # メモファイル出力
        print OUT $line,"¥n";
}
# カテゴリデータの処理
if($line =~ /^C:¥s+(.*)$/i){
        $category = $1;# カテゴリデータを取得する
        next unless $category;# カテゴリデータがない場合はスキップ
        # カテゴリとタイトルデータが揃ったら処理をする
        # タイトルデータが空の場合はスキップ
        if($title ne ""){
                $catid = time + $catcount;
                $articles{"${category}_$catid"} = $title;
                # CGI 出力
                print "<div class=¥"emph¥">¥n";
                print "<A HREF=¥"/$memodir/memo_index.html#$category¥">";
                print "[$categories{$category}]</A> ";
                print "<A NAME=¥"${category}_$catid¥">$title</A></div>¥n";
                $title = "";
                $category = "";
                $catcount++;
        }
# タイトルデータの処理
}elsif($line =~ /^T:¥s+(.*)$/i){
        $title = $1;# タイトルデータを取得する
        next unless $title;# タイトルデータがない場合はスキップ
        # タイトルにハイパーリンクを張る場合の処理
        $title =~ s/¥[U:¥s+([^;]+);T:¥s+([^¥]]+)¥]/<a href="$1">$2<¥/a>/i;
        $title =~ s/¥[U:¥s+([^;¥]]+)¥]/<a href="$1">$1<¥/a>/i;
        # カテゴリとタイトルデータが揃ったら処理をする
        # カテゴリデータが空の場合はスキップ
        if($category ne ""){
                $catid = time + $catcount;
                $articles{"${category}_$catid"} = $title;
                # CGI 出力
                print "<div class=¥"emph¥">¥n";
                print "<A HREF=¥"/$memodir/memo_index.html#$category¥">";
                print "[$categories{$category}]</A> ";
                print "<A NAME=¥"${category}_$catid¥">$title</A></div>¥n";
                $category = "";
                $title = "";
                $catcount++;
        }
# パラグラフ（段落）データの処理
}elsif($line =~ /^(?:DE:|P:)¥s+(.*)$/i){
        $line = $1;
        next unless $line;# データが空の場合はスキップ
        &linking($line);
        print "<p>$line</p>¥n";# CGI 出力
```

```
        next;
# 整形済みテキストデータの処理 (1)
}elsif($line =~ /^PRE:$/i){
        print "<pre>¥n";# CGI 出力
        $pre = 1;# フラグを立てる
        next;
# 整形済みテキストデータの処理 (2)
}elsif($line =~ /^¥/PRE:/i){
        print "</pre><br>¥n";# CGI 出力
        $pre = 0;# フラグを元に戻す
        next;
# 番号なしリストデータの処理 (1)
}elsif($line =~ /^UL:$/i){
        print "<ul>¥n";# CGI 出力
        $ul = 1;# フラグを立てる
        next;
# 番号なしリストデータの処理 (2)
}elsif($line =~ /^¥/UL:$/i){
        print "</ul><br>¥n";# CGI 出力
        $ul = 0;# フラグを元に戻す
        next;
# 番号付きリストデータの処理 (1)
}elsif($line =~ /^OL:$/i){
        print "<ol>¥n";# CGI 出力
        $ol = 1;# フラグを立てる
        next;
# 番号付きリストデータの処理 (2)
}elsif($line =~ /^¥/OL/i){
        print "</ol><br>¥n";# CGI 出力
        $ol = 0;# フラグを元に戻す
        next;
# 階層型リストデータの処理 (1)
}elsif($line =~ /^ULS:$/i){
        $blevel = 0;
        print "<ul>¥n";# CGI 出力
        $uls = 1;# フラグを立てる
        next;
# 階層型リストデータの処理 (2)
}elsif($line =~ /^¥/ULS:$/i){
        while($blevel > 0){
                print "¥t" x $blevel;# CGI 出力
                print "</ul>¥n";
                $blevel--;
        }
        print "</ul><br>¥n";# CGI 出力
        $uls = 0;# フラグを元に戻す
        next;
# 定義リストデータの処理 (1)
```

```
		}elsif($line =~ /^DL:$/i){
			# <dl>タグの前にタブを入れて、メモ開始の<dl>タグと区別
			print "¥t<dl>¥n";# CGI 出力
			$dl = 1;# フラグを立てる
			next;
		# 定義リストデータの処理 (2)
		}elsif($line =~ /^¥/DL/i){
			# </dl>タグの前にタブを入れて、メモ終了の</dl>タグと区別
			print "¥t</dl><br>¥n";# CGI 出力
			$dl = 0;# フラグを元に戻す
			next;
		# 表データの処理 (1)
		}elsif($line =~ /^ta:$/i){
			$ta = 1;# フラグを立てる
			$count = 0;
			next;
		# 表データの処理 (2)
		}elsif($line =~ /^¥/ta:$/i){
			print "</TABLE><BR>¥n";# CGI 出力
			$ta = 0;# フラグを元に戻す
			next;
		}elsif($line =~ /^bq:$/i && $pre != 1){
			$bq = 1;
			print "<BLOCKQUOTE>¥n";
			next;
		}elsif($line =~ /^¥/bq:$/i && $pre != 1){
			$bq = 0;
			print "</BLOCKQUOTE>¥n";
			next;
		# 改行データの処理 (1)
		}elsif($line =~ /^br:$/i){
			$br = 1;# フラグを立てる
			next;
		# 改行データの処理 (2)
		}elsif($line =~ /^¥/br:$/i){
			$br = 0;# フラグを元に戻す
			print "<br>¥n";# CGI 出力
			next;
		# memol 言語で記述した画像データの処理
		}elsif($line =~ /^¥[[^¥]]+¥]$/ && $pre != 1 && $ul != 1 && $ol != 1 && $uls
!= 1 && $dl != 1 && $ta != 1 && $br != 1 && $bq != 1){
			&linking($line);
			print $line, "¥n";# CGI 出力
		# その他の行の処理
		}elsif($line !~ /^¥w{1,3}:$/ && $line !~ /^¥w{1,3}: .+$/ && $pre != 1 &&
$ul !=1 && $ol != 1 && $uls != 1 && $dl != 1 && $ta != 1 && $br != 1 && $bq != 1){
			&linking($line);
			print $line, "¥n";# CGI 出力
```

```
}
# フラグが立っている状態でのデータ処理
# 整形済みテキストデータの処理 (3)
# <>記号のみ実体参照エンコードする
if($bq == 1){
        &linking($line);
        print "$line¥n";
}
if($pre == 1){
        $line = &entities_encode($line);
        print $line, "¥n";# CGI 出力
}
# 番号なしリストデータの処理 (3)
if($ul == 1){
        &linking($line);
        print "<li>$line¥n";# CGI 出力
}
# 番号付きリストデータの処理 (3)
if($ol == 1){
        &linking($line);
        print "<li>$line¥n";# CGI 出力
}
# 階層型リストデータの処理 (3)
if($uls == 1){
        &linking($line);
        if($line =~ /^([ ¥t]*)[^ ¥t].*$/){
                $level = length $1;
                if($blevel > $level){
                        $diff = $blevel - $level;
                        if($diff >= 1){
                                for($i = 1;$i<=$diff;$i++){
                                        $ts = "¥t" x ($blevel - $i + 1);
                                        print "$ts</ul>¥n";# CGI 出力
                                }
                        }
                        $line =~ s/^([ ¥t]*)([^ ¥t].*)$/$1<li>$2/;
                }elsif($blevel == $level){
                        $line =~ s/^([ ¥t]*)([^ ¥t].*)$/$1<li>$2/;
                }elsif($blevel < $level){
                        $diff = $level - $blevel;
                        if($diff >= 1){
                                for($i = 1;$i<=$diff;$i++){
                                        $ts = "¥t" x ($blevel + $i);
                                        print "$ts<ul>¥n";# CGI 出力
                                }
                        }
                        $line =~ s/^([ ¥t]*)([^ ¥t].*)$/$1<li>$2/;
                }
```

```
                }
                $blevel = $level;
                print $line,"¥n";# CGI 出力
        }
        # 定義リストデータの処理 (3)
        if($dl == 1){
                if($line =~ /^([^:]+:) (.*)$/){
                        $prop = $1;
                        $str = &linking($2);
                        print "¥t¥t<dt>$prop¥n¥t¥t<dd>$str¥n";# CGI 出力
                }
        }
        # 改行データの処理 (3)
        if($br == 1){
                &linking($line);
                $line =~ s/^(.*)$/$1<br>/;
                print $line,"¥n";# CGI 出力
        }
        # 表データの処理 (3)
        if($ta == 1){
                $count++;
                if($count == 1){
                        $caption = &linking($line);
                        # CGI 出力
                        print "<TABLE BORDER=7>¥n";
                        print "<CAPTION ALIGN=top><B>$caption</CAPTION>¥n";
                }elsif($count == 2){
                        # 表の項目データを取得して、HTML を出力します。
                        &linking($line);
                        @heads = split(/:/, $line);
                        print "<TH>";# CGI 出力
                        for($i=0; $i<$#heads; $i++){
                                print "$heads[$i]</TH><TH>";# CGI 出力
                        }
                        print "$heads[$#heads]</TH>¥n";# CGI 出力
                }else{
                        # 規則的なテキストデータから１行ずつ変換して出力します。
                        &linking($line);
                        @fd = split(/:/, $line);
                        print "<TR>¥n<TD>";# CGI 出力
                        for($i=0; $i<$#fd; $i++){
                                print "$fd[$i]</TD><TD>";# CGI 出力
                        }
                        print "$fd[$#fd]</TD>¥n</TR>¥n";# CGI 出力
                }
        }
    }
}
```

```
# メモ HTML 最終部分の CGI 出力
print "</DL>¥n<HR>¥n";

# メモ作成・編集用フォームの CGI 出力
print "<FORM action=¥"$cgidir/wemo_edit.cgi¥">¥n";
# <<重要>> INPUT タグの hidden 属性で、メモの日付 date を $date として
# wemo_edit.cgi に渡すことによって、どのメモファイルを処理するのかが同定される
print "<INPUT TYPE=¥"hidden¥" NAME=¥"date¥" VALUE=¥"$date¥">¥n";
# SELECT タグによるカテゴリの選択メニューを出力する
print "<SELECT name=¥"category¥" size=1>¥n";
print "<OPTION selected></OPTION>¥n";

# OPTION タグで、カテゴリの選択メニューを作成する
foreach $catstr (sort keys(%categories)){
        print "<OPTION value=¥"$catstr¥">$categories{$catstr}</OPTION>¥n";
}
# フォームの完成
print "</SELECT><INPUT TYPE=¥"submit¥" VALUE=¥"作成・編集¥"></FORM>¥n";

print "[ <a href=¥"$cgidir/wemo_list.cgi¥">Wemo list</a> ]¥n";

# CGI 出力の終了
print "</BODY>¥n</HTML>¥n";

close(OUT);

# 文字データ中の wemo 言語文字列の処理
sub linking{
        $catid = time + $catcount;# アンカー名を確実に区別して生成する

        # ハイパーリンクの処理
        # [u: URL;T: タイトル]
        $_[0] =~ s/¥[U:¥s+([^;]+);T:¥s+([^¥]]+)¥]/<a href="$1">$2<¥/a>/gi;
        # [u: アンカーベース名;t タイトル]
        $_[0] =~ s/¥[N:¥s+([^;]+);T:¥s+([^¥]]+)¥]/<a name="${1}_$catid">$2<¥/a>/i;
        # [u: URL]
        $_[0] =~ s/¥[U:¥s+([^;¥]]+)¥]/<a href="$1">$1<¥/a>/gi;

        # 画像データの処理
        # [i: 画像 URL]
        $_[0] =~ s/¥[I:¥s+([^;¥]]+)¥]/<IMG SRC="$1" ALIGN=LEFT HSPACE=20 ALT="$1">/gi;
        # [i: 画像 URL;t: タイトル]
        $_[0] =~ s/^¥[I:¥s+([^;]+);T:¥s+([^¥]]+)¥]$/<IMG SRC="$1" ALIGN=LEFT HSPACE=20
ALT="$1">$2<BR CLEAR=ALL>/i;
        # [i: 画像 URL;u: 画像にリンクする URL;t: タイトル]
        $_[0] =~ s/^¥[I:¥s+([^;]+);U:¥s+([^;]+);T:¥s+([^¥]]+)¥]$/<IMG SRC="$1" ALIGN=LEFT
HSPACE=20 ALT="$1"><a href="$2">$3<¥/a><BR CLEAR=ALL>/i;
```

```
        # [u: 画像にリンクするURL;i: 画像URL;t: タイトル]
        $_[0] =~ s/^¥[U:¥s+([^;]+);I:¥s+([^;]+);T:¥s+([^¥]]+)¥]$/<a href="$1"><IMG SRC="$2"
ALIGN=LEFT HSPACE=20 ALT="$2"><¥/a>$3<BR CLEAR=ALL>/i;
        # [u: 画像にリンクするURL;i: 画像URL;u: タイトルにリンクするURL;t: タイトル]
        $_[0] =~ s/^¥[U:¥s+([^;]+);I:¥s+([^;]+);U:¥s+([^;]+);T:¥s+([^¥]]+)¥]$/<a
href="$1"><IMG SRC="$2" ALIGN=LEFT HSPACE=20 ALT="$2"><¥/a><a href="$3">$4<¥/a><BR
CLEAR=ALL>/i;

        # 『』で囲まれた部分を引用部として処理する
        $_[0] =~ s/『([^』]+)』/『<cite>$1<¥/cite>』/ig;

        # URL 文字列に URL 文字列の表すハイパーリンクを張る
        # URL 文字列の次の文字が URL 文字列を構成する文字列以外であり、
        # かつマークアップ記号ではない場合、
        $_[0] =~ s/(http:¥/¥/[#%a-z0-9¥-_.!~*'¥(¥);¥/?:@&=+¥$,]+)([^<>"#%a-z0-9¥-_.!~*'¥
(¥);¥/?:@&=+¥$,]+)/<a href="$1">$1<¥/a>$2/ig;
        # URL 文字列の並びの次が行末である場合、
        $_[0] =~ s/(http:¥/¥/[#%a-z0-9¥-_.!~*'¥(¥);¥/¥?:@&=+¥$,]+)$/<a href="$1">$1<¥/a>/i;

        $catcount++;
        return $_[0];
}

# 現在の日付を yyyy-mm-dd 形式で得る
sub date{
        ($mon = (localtime)[4] + 1) =~ s/^(¥d)$/0$1/;
        ($mday = (localtime)[3]) =~ s/^(¥d)$/0$1/;
        $year = (localtime)[5] + 1900;
        return $year . "-" . $mon . "-" . $mday;
}

# <>実体参照エンコード（<pre>タグの中で使う）
sub entities_encode{
  my($str) = @_;
  $str =~ s/</&lt;/g;
  $str =~ s/>/&gt;/g;
  return $str;
}

# URL エンコーディング
sub jurl_encode{
  my($str) = @_;
  $str =~ s/([^a-z0-9¥-_.!*'¥(¥)~ ])/sprintf("%%%02X", ord($1))/egi;
  $str =~ s/ /+/g;
  return $str;
}

__END__
```

編集後記

さて、ようやく第2号の編集作業も終わりに近づいてきた。創刊号と同じ繰り返しの謝辞になるが、繰り返しておきたい。本会誌は、OpenOffice.org を使わせてもらって、編集している。素晴らしいツールである。このようなツールがあるおかげで、アマチュアが電子出版を容易に行うことができる。スクリプティング言語などを含め、有用なコンピューティングツールがフリーかつオープンに提供されるようになったことは大変ありがたいことである。フリーソフトウェアの関係者に改めて感謝したい。

なんとか第2号を出せる。今号から海鳥さんのマンガを掲載できることになったのは大変うれしい。TSNET にさらに多くの人に参加していただいて、TSNET スクリプト通信を継続して刊行できるようになれば素晴らしいと思う。次回は秋も深まる 11 月の刊行を予定している。是非、ご参加を^^)/~

2008-08-16
jscripter

ＴＳＮＥＴスクリプト通信

2008 年 8 月 19 日　　1.2.004 版発行

投稿規程

TSNETWiki ： 「投稿規程」のページを参照のこと

編集委員会(投稿順)

機械伯爵	kikwai at livedoor dot com
Ｙさ	saw at mf-nokuchi2pho dot ne dot jp
海鳥	kaityo256 at nifty dot ne dot jp
jscripter	jscripter9 at gmail dot com

著作権

使用許諾・配布条件

免責事項

編集ソフトウェア

OpenOffice.org 2.3.1 Writer
PDF InfoMaker 1.20

発行所(一次配布所)

TSNETWiki ： 「TSNET スクリプト通信」のページを参照のこと
