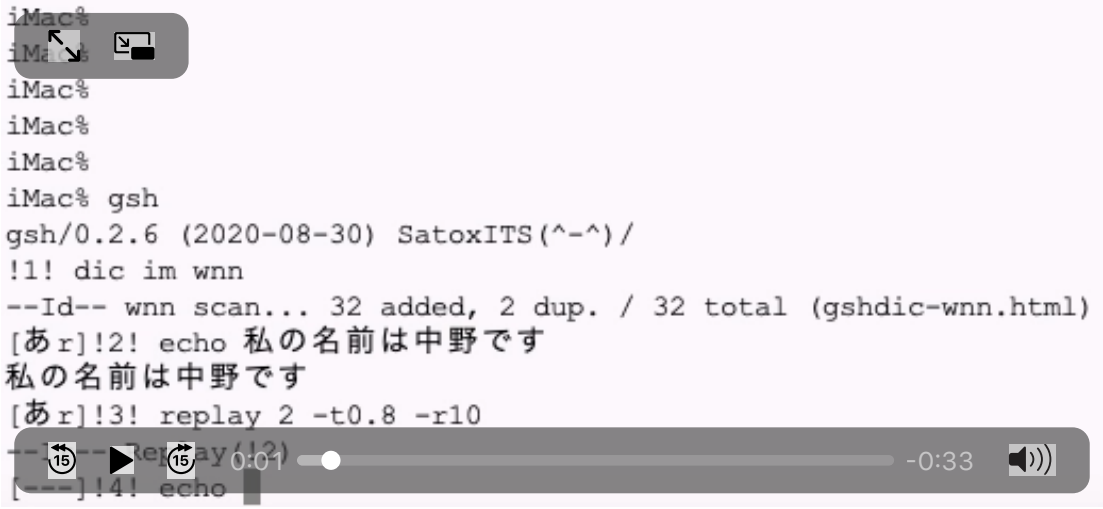
株式会社 ITS MORE
2020年4月設立

2020年8月30日 投稿者: SATOXITS

GShell 0.2.6 – 私の名前は中野です



このキーボード入力は人間業ではありません

社長：乗りかかった船でもあるので、GShell IMEの件はもう少しキリの良いところまで進めて置きたいですね。

開発：昨日の入力リプレイ機能は、自分の本業系の脳機能を使ったので少し疲れたように思います。IME の件と比べると、with fun が弱い。

社長：我々のIMEの原点はやはりWnnだと思うわけです。当時はIMEというような言葉があった記憶は無いですけど。

基盤：MS あたりの造語ですかね？

開発：それ以前はパソコンで松 vs 一太郎の時代でしたね。PC98で。

社長：まあ私らはvi育ちは当然、松派でしたけどね。

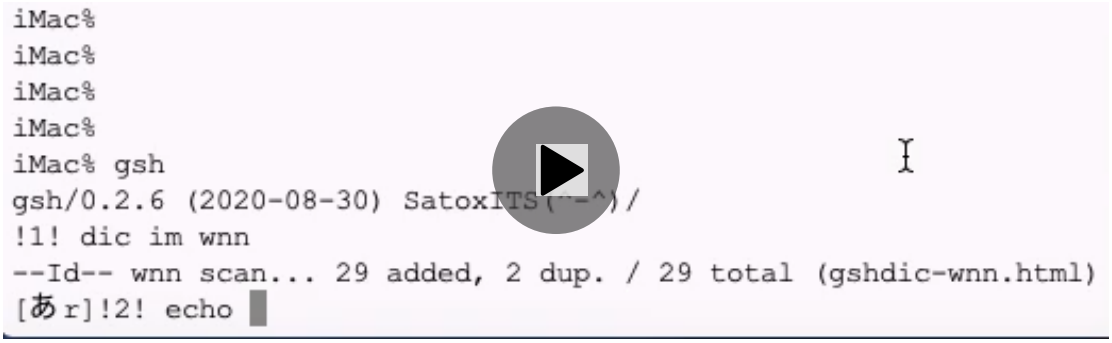
開発：最初は5インチフロッピーで、最後は3.5インチになりましたね。そうこうするうちにパソコン端末時代が終わってUnixワークステーション時代になり、Wnn になり、Canna もでてきた。

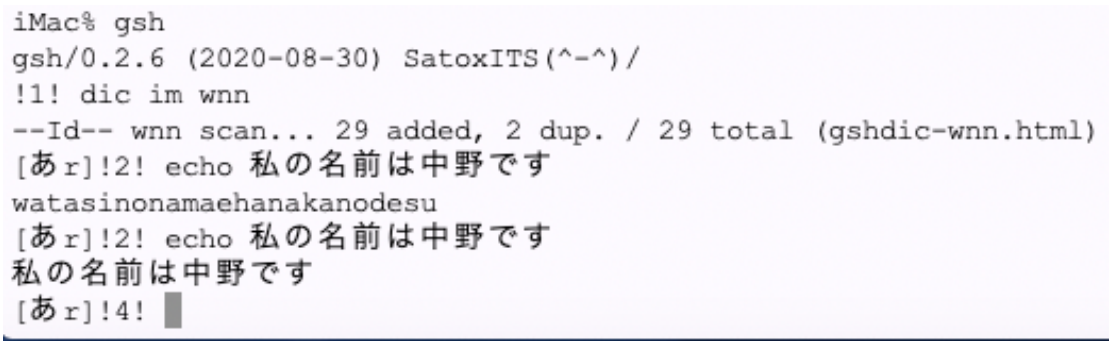
社長：Cannaの中の人たちとはOnew MLの関係で1994年か95年に東京でオフミをやりました。みなさんどうしてますかね。

開発：Canna懐かしス。

＊　＊　＊

開発：さて、それで Wnn と言えば「わたしのなまえはなかのです」なわけです。ちょっと未来から来ました。最初に、下で色々検討する方式の、目指すところ、結論を示したいと思います。自動表示変換モード（¥j）＋手動書き換え（¥i）でやります。





社長：史上最もなめらかな「私の名前は中野です」かも知れませんね。

基盤：キータッチがもっと速ければｗ

開発：なにせ、人間は変換用の指示は何も出しておらず、単にローマ字表記の列をべた入力している。随時、GShell IME は入力を変換して表示している。それだけです。

開発：舞台裏。辞書はこんなふうになっています。

```
iMac% gsh
gsh/0.2.6 (2020-08-30) SatoxITS(^-^)/
!1! dic im wnn
--Id-- wnn scan... 29 added, 2 dup. / 29 total (gshdic-wnn.html)
[あr]!2! echo 私の名前は中野です
watasinonamaehanakanodesu
[あr]!2! echo 私の名前は中野です
私の名前は中野です
[あr]!4! dic im -v wnn
--Id-- ReadDic(412,<nil>)
--Id-- wnn gshdic-wnn.html
<meta charset="UTF-8">
<textarea cols=80 rows=40>
//dicver        GShell\sIME\sdictionary\sfor\sWnn\s//\s2020-0830
GShell  GShell
わたし  私
watashi 私
watasi  私
なまえ  名前
namae   名前
なかの  中野
nakano  中野
wa      わ
ta      た
si      し
shi     し
no      の
na      な
ma      ま
e       え
ha      は
na      な
ka      か
no      の
de      で
su      す
e\s     echo
echo    echo
dt      date\s+'%Y%m%d-%H:%M:%S'
tion    tion
%t      %t      // to be an action
</textarea>

--Id-- wnn scan...
--Id-- wnn scan... 0 added, 31 dup. / 29 total (gshdic-wnn.html)
[あr]!5!
```

開発：で、こういうべたな辞書方式でGo！という結論に、下記では至るわけです。少なくとも昼食前の時点では。

＊　＊　＊

開発：さて、一旦ベースに立ち戻ります。

```
iMac% gsh
gsh/0.2.6 (2020-08-30) SatoxITS(^-^)/
!1! dic im wnn
--Id-- wnn scan... 25 added, 2 dup. / 25 total (gshdic-wnn.html)
!2! echo watasinonamaehanakanodesu
watasinonamaehanakanodesu
[あr]!2! echo わたしのなまえはなかのです
watasinonamaehanakanodesu
[あr]!3! echo 私の名前は中野です
わたしのなまえはなかのです
[あr]!4! echo 私の名前は中野です
私の名前は中野です
```

開発：!2 から表示だけ変換、オリジナルの文字列は変えないモードに入っています。!3,4,5 はヒストリ機能というか上向き矢印で!2のコマンド行を呼び出して手動で「追加変換」をしています。

基盤：段階が複雑過ぎると思います。

社長：あと、表示されている状態が実際の結果にならないというのは直感に反しますね。

開発：実装上、根っこにこういう機能というか状態は必要なので。ユーザには、それらを使いやすい組み合わせにパッケージした状態で提供すれば良いと思います。逆変換は難しいと思いますから、なるべくオリジナルの文字列を保持したい。

社長：オリジナル入力列の保持は技術的には面白いと思いますが、単に日本語を入力したい場合には邪魔かなと思います。

基盤：ヒストリを取っておいてundoで戻れば良いように思います。

開発：そのへんは、両立する内部＋外部表現形式があれば良いのかなとも思います。そ

こにある文字列だけで状態を全て持っているというのはどうしても魅力です。例えば変換モード自体を内蔵する文字列みたいな。

開発：次に、自動書き換えつまり入力文字列の自動置き換えモード。

社長：入力を置き換えるというのは、通常の言語処理系にはない方式でしょうね。

開発：そこもまた、IMEが古典的な言語処理系で実現されてない理由の一つかなと思います。しかも、ローマ字→かな→漢字という2段階の書き換えを行っている。

社長：そこはまとめて、ローマ字→（かな|漢字）という辞書を作ればよいのではないですかね。

開発：そこはにはまず、ローマ字→かな変換が一意でない、つまりローマ字表記とヘボン式と訓令式を受容したいという問題があります。たとえば私は「ち」語の先頭では「chi」と打つことが多いですが、語の中では「ti」と売ってることが結構多いようなんです。

社長：かなへの変換を想定しないローマ字表記ではヘボン式に親しんでますからね。

基盤：効率を考えれば訓令式「ti」ですよね。

開発：ローマ字レベルで内部表現を訓令式に正規化してしまうという手はあるとは思います。なんにしても、「 chikara || tikara > 力」というような重複した辞書を持つものはいかがなものかと。まあ、ローマ字読み状態も保持したいと思うと、これも致し方ないとは思いますが。

社長：そういう意味で、ひらがな表記の辞書だと、ひらがな段階で正規化されるわけですね。「chi ka ra || ti ka ra > ち か ら > 力」という。

基盤：変換辞書のためのメモリもストレージもほぼ無尽蔵にあるんですから、重複とか気にしなくて良いのでは。

開発：そうですね…　ただ、多段方式での変換には、ひらがな表記にした段階で一旦「確定する」という機能というか効能もあるわけです。たとえば「ち」という1つの文字に確定できる。これを「ti」や「chi」の状態で残しておくと、入力済の部分にカーソ

ル移動したときに「t | i > tい」とか「c | hi > cひ」とか「ch | i > chい」とか、面白いけどうざいことになるわけです。

社長：昔に戻ってそこを直したいことはまず無い。カーソル移動の効率も悪いですね。

開発：ところで上で書いた、カーソル位置を示す「|」は、保存する文字列にも残しておくと面白いかもですね。再編集するときに、どこから編集を再開するかを指示できる。

社長：まあ「|」はパイプやORを始め、あまりによく使われてますから、やるとしたら「|||」とか、「¥…」あたりでしょうね。

開発：私はGoで「…」という予約語が使われてるのを見て、最初ぎょっとしましたw

社長：ともかく、「世界辞書」で変換が気持ち良いのは、「se > せ」「 ka > か」「i > い」 とは別に「sekai > 世界」 とい入力が定義されていて、s、せ、せk、せか、から直に 世界に変わってくれるからだと思います。一旦「せかい」になってから「世界」へという段階が無いのが気持ち良い。

基盤：「echo > えcほ || えちょ」にならないように、「echo > echo」という定義もあると良いと思います。

社長：「echo > エコー || おうむがえし」という定義もあって、どっちか選ぶというのも良いかもですね。まあコマンド入力の先頭に出てくるのは echo に決まってますから、文脈で決めれば良いことですが。

基盤：「エコー」とか「おうむがえし」という日本語のコマンドがあっても悪くないですよね。そもそも内部的に echo という文字列が保持されてるなら、一貫して echo なんでしょうけど。

開発：日本語英語交じりの文章、とくに技術文書はそうですが、には英単語の辞書は絶対必要だと思いますね。これは英単語だから、英単語として表示するという。そもそも前後に空白のある文字列は、日本語に変換するよりローマ字のまま表示する方を優先するとかでも良いですが。

開発：実装上は、辞書による書き換えを、変換時に多段階に再帰的に適用するか、あらかじめ一段階の辞書に展開してしまって、いわゆるローマ字かな変換とかな漢字変換を

一体化してしまうか、という選択肢があります。前者はあたりまえの技法なので、面白そうな後者をやってみたい気がします。

開発：多段階の変換をしたい場合には、たとえばハードコーディングされたアルゴリズムとして存在するより、辞書というデータを複数に分けておいて、どの辞書をどういう順序で適用するかという形で一般化したほうが面白いと思うのです。これなら何段階でもありえます。まあ普通に言えば、ローマ字かな辞書と、かな漢字辞書なわけですが。

社長：辞書には普通は品詞情報も持ってたりするわけですよね。

開発：まったく日本語の、というか個別の言語の知識無しでできると面白いかと。できる限りシンプルに、ナイーブに、ブルートフォース的に。

社長：なんにしても、いろんな制約とか、他の面にいろいろ弊害は出ると思いますが、「変換」という人間の指示なしで、本来のコンテンツの情報を入力するだけで結果が出るというのは、あまりに魅力的です。

開発：インクリメンタルに変換して、人間が「それで良い」と思ったその場で確定して、その先に進むという入力形態になるんだと思います。

社長：長く入力しておいてあとで変換を確定する式があるのは、確定する操作が面倒にできているからかも知れないですね。

基盤：複数の候補がある時には、候補の数を表示するとか、色で知らせしてくれるとよいですね。

社長：なんにしても、既に入力済みの情報だけから出力を作るのであれば、原理的にはそれができるはずですよね。

開発：ちょっとやってみたんですが、やはり難関は「すももももももものうち」かなと思うんです。これは人間にしか切れ目がわからない。Google IMEにはこれが一つの語として登録されていて一発変換できるわけですが、それはある意味ズルです。そもそもそれなら、「すももも > すももも桃も桃の内」って入れときたいくらいです。これを実直なMS の IME で一括変換しようとすると、ひどいことになる。単語、文節毎に確定して行くほうがずっとよいわけです。ここをなんとか、なめらかに入力できないだろうかと。

社長：難題ですね。面白い。お昼食べてきます。

＊　＊　＊

基盤：結局食事に行きませんでしたね。

社長：面倒くさくなって。そもそも今時点、とうもろこし以上に魅力な外食が無いのです。

レンジ：ひぺぱ、ぴぺぱ、…

開発：出来ました。うん、思ったとおり細身だと甘みが強いですね。もぐもぐ。

社長：もぐもぐ。若いうちは糖分とデンプンで、成熟したらタンパクになるとかですかね。

開発：加熱すると甘くなる化学構造かどうかもあるんじゃないですかね。もぐもぐ。

基盤：あー、美味しかったー。

開発：いやー、このBGM、もう20回転くらいしましたが、全然飽きないですね。もっと聴きたくなるくらいな。

社長：BGMガンガンかけて仕事できる職場ってのも夢のひとつでしたね。

開発：ヘッドホンとかイヤホンて気持ち悪いですよね。音も自然には広がらないし。

＊　＊　＊

社長：ところで思ったんですが、入力列を replay するときに実際のスピードで再生するだけでなく、早送りとか、おそ回しとか、定間隔にしちゃうとかいうオプションあると、面白いんじゃないですかね。

開発：そうですね。それは簡単。

基盤：逆回しもあると良いですね。ローテートシフトするとか。

社長：それは面白すぎる。

開発：何か実用の役に立つ機能にならないですかね…

社長：rot13とかは欲しいですよね。

＊　＊　＊

社長：どんな感じでしょう？

開発：それが、昨日リクエストのあった、replay時に変換モードを再現する件を実装したのですが、そこで考え事を。これは当然、モード情報を特殊な、イベントの一種として送ってやるのが自然だと思います。バイト列というかオクテットストリームのインバウンドで。で、どういう形式にしようかと。自然と 0xFF で始まるバイト列になったわけです。で、ふとこれは、前に見た事があるなと。そう、Telnet プロトコルですよ。

社長：なるほど…　Telnetに合わせておけば、GShellをデータストリーム中継型のリモートシェルとして使う時に、環境変数とか端末情報の伝達がそのまま使えますね。

開発：そうなんですが、まてよと。そのオクテットストリームをTelnetを理解しない普通の受信者が受け取った時はどうだろうかと。

社長：まあ、受け取った文字列がTelnetのパケットだとは思うはずないので、びっくりするでしょうね。化け化けでしょう。

開発：バイナリデータはどの道化けるとは思うんですが、UTF-8で無いのが嫌だなと思ったわけです。なので、Telnet のパケットを UTF-8でエンコードしてはどうかと。

開発：UTF-8にするメリットは、可変長のイベント情報をUTF-8で長さ規定して送れることです。

基盤：Unicodeの面白い文字になるようなパケットを作ると面白いかもですね。

社長：「表示されない文字」というUnicodeがあれば、invisibleにできますね。お豆腐にもならない。まあそのへんは、ANSIシーケンスのほうが柔軟だとは思います。意味のない機能シーケンスを送るとか。

開発：ISO-2022 コードも使えるかもですねｗ

社長：実際私は MIMEで、文字コード切り替えコードを続けて送って状態伝達をする方法を提案したりしてました。

基盤：モールス符号みたいですねｗ

開発：入力列として伝達するんで、visibleに送ってBSで消すという技もアリですかねｗ

開発：ああそれで、早回し再生の件はこんなふうになりました。

社長：これ、リプレイのために送信しているイベントを別ウィンドウで表示すると良いかもですね。

基盤：というか、IMEでの変換候補とか、ヒストリのビューなんかも別ウィンドウになるべきではないでしょうか。

開発：そのへんは、GShellがHTTPサーバになって、HTTPでブラウザに表示すれば良いかなとも思っていたのですが…　まあ変換の候補を表示するには専用のポップアップ的ウィンドウがあったほうが良いとは思います。でもANSIシーケンスでウィンドウもどきを端末に表示するのも面白いかなと。

社長：curses的なものを使うかどうかですね。

開発：使う場合には使い倒す。使わないなら一切使わない、というところかと思います。なんせ実際現在、全く使っていません。画面への転送と表示が速いもんで、キー入

力ごとに変換してばかちょんで行の全部を書き直してます。このIME、JavaScriptでDOM上に書く時にも使いたいと思っていまして、その場合にはANSIが使えるとは思えないと言うこともあります。

＊　＊　＊

開発：一応すももも課題もクリアしました。

```
iMac% gsh
gsh/0.2.6 (2020-08-30) SatoxITS(^-^)/
!1! dic im sumomo
--Id-- sumomo scan... 18 added, 0 dup. / 18 total (gshdic-sumomo.html)
[あr]!2! echo すももも桃も桃の内
sumomomomomomomomomonouti
[あr]!3!
```

基盤：なんかズルしてませんかｗ

開発：すももも専用の辞書ですから。

社長：文脈に最適な辞書を選択するのは人として当然です。

＊　＊　＊

社長：だだいまー。

経理：だいぶ飲んでますね。

社長：今日はあの、謎の定食屋？中華料理？でした。いつものようにマスターは寝そべってテレビ見てましたけどｗ。でもあそこの出色は刺し身なんですよね。下手な和食屋よりハイグレード。でもって会計はほぼ原価なんです。まかないですかってｗ

社長：ああそれで、飲みながら思ったんですが、要するに入力は**分かち書きすればいいんじゃね？**ということです。日本語の変換を面倒にしている理由の一つは、これが分かち書きの言語でないからです。最終結果として分かち書き表示された日本語というのはやはり変だと思いますが、入力を分かち書きするのは別に何の問題もない。

開発：処理系としては分解処理がなくて楽ですが、そもそもユーザが分かち書き入力の手間を許容するかどうかですね。

社長：単純なそれをしないかわりに、逆にすごく面倒な操作が増えているんじゃないかと思うわけです。

基盤：スペースキーは重いので、これで分かち書きはあまりうれしくないですね。

社長：私はカンマを希望。

開発：常にcsv入力みたいなｗ

— 2020-0830 SatoxITS

http-im3-gsh-gsh-0.2.6.go_ ダウンロード

```
/*
                                    GShell version 0.2.6 // 2020-08-30 // SatoxITS
```

GShell // a General purpose Shell built on the top of Golang

```
It is a shell for myself, by myself, of myself. —SatoxITS(^-^)

0 | | Fork | Stop | Unfold | */ /*
▼ Statement
```

Fun to create a shell

For a programmer, it must be far easy and fun to create his own simple shell rightly fitting to his favor and necessities, than learning existing shells with complex full features that he never use. I, as one of programmers, am writing this tiny shell for my own real needs, totally from scratch, with fun.

For a programmer, it is fun to learn new computer languages. For long years before writing this software, I had been specialized to C and early HTML2 :-). Now writing this software, I'm learning Go language, HTML5, JavaScript and CSS on demand as a novice of these, with fun.

This single file "gsh.go", that is executable by Go, contains all of the code written in Go. Also it can be displayed as "gsh.go.html" by browsers. It is a standalone HTML file that works as the viewer of the code of itself, and as the "home page" of this software.

Because this HTML file is a Go program, you may run it as a real shell program on your computer. But you must be aware that this program is written under situation like above. Needless to say, there is no warranty for this program in any means.

Aug 2020, SatoxITS (sato@its-more.jp)

```
*/ /*
```

▼ Index

```
Documents
        Command summary
Go lang part
        Package structures
                import
                struct
        Main functions
                str-expansion   // macro processor
                finder          // builtin find + du
                grep            // builtin grep + wc + cksum + …
                plugin          // plugin commands
                system          // external commands
```

```
                builtin         // builtin commands
                network         // socket handler
                remote-sh       // remote shell
                redirect        // StdIn/Out redireciton
                history         // command history
                rusage          // resouce usage
                encode          // encode / decode
                IME             // command line IME
                getline         // line editor
                scanf           // string decomposer
                interpreter     // command interpreter
                main

JavaScript part
        Source
        Builtin data
CSS part
        Source
References
        Internal
        External
Whole parts
        Source
        Download
        Dump


*/ //
```

Go Source

```
//
```

Considerations

```
// /*
```

References

```
*/ /*
```

Raw Source

```
*/ /*
```

```
--> *///
```