Tree Shell

-もう一つのデスクトップアプローチ-

峠 裕

2003年8月6日

概要

Linuxでも本格的なデスクトップ環境やGUIベースのアプリケーションが充実してきている昨今ですが、なかなか重厚な作りのものが多いようで、古き良き(?)コマンドライン文化との連続性も希薄なようだし、ちょっと寂しく感じるときもあります。そこで、既存のコマンドを再利用するような簡単な実装でユーザフレンドリな環境は出来ないものかと思い、少し工夫してみました。本稿では、実装したプログラムを例にとって、どのようなアプローチをしたかなどについて紹介してみたいと思います。

1 はじめに - アプローチの前提 -

一口にデスクトップアプローチといっても、いろいろと切口がありますが、今回意識したポイントは

1. 実際のユーザ層を意識して考えてみる

   Linuxをデスクトップ OSとして利用する「実際のユーザ」として、現時点では Windowsや Macからの乗り換えユーザを意識すべきでしょうし、仮にはじめてコンピュータに触る人の（デスクトップ）OSが Linuxだっとしても、いわゆる ITのプロではないユーザ層が大部分を占めるでしょうから、それにふさわしいインターフェースを提供できることが必要になってくると思います。Windowsや Macは OSの持つ「全て」の機能をGUIベースで提供出来ていますが、それらと同等の効用を Linux上で得ようとするとコマンドラインベースのスキルが必要な状況が依然続いているかと思います。

2. ネットワーククライアントとしての側面を考慮する

   デスクトップ OSとしての現実的な利用シーンを考えると、（職場などの）LAN上でのクライアントノードとしての振舞いは結構切実だと思います。Windowsネットワークのファイル共有サービスなどに簡単にアクセスできるインタフェースが提供されていないと、業務に支障の出る恐れもあるのではないでしょうか。

3. 既存の資産・文化との連続性を確保できるような工夫をする

   パイプや正規表現などを駆使して既存のソフトウェア資産を柔軟に活用できるコマンドライン文化は、Linuxなどの Unixライクな OSの魅力の一つだと思いますし、既にあるコマンドを GUI化するために個々に大作アプリを作成するのはちょっと悲しい気がします。

くらいでしょうか。ともかくも簡単なツールを作ってみましたので、その機能や設計・実装（のコンセプト）などを順を追って見ていきたいと思います。

2　ツール概要

2.1　イメージ

早速ですが、作成したツールのイメージを図 1〜図 3 に示します。画面が途切れていますが、これは 1 アプリケーションでの部分描写になります。一つのアプリケーションで samba や ftp などのネットワーククライアントとして GUI でのアクセスを提供しているので、「1.」「2.」（p.1）のアプローチには多少の解答になっているかと思います。しかし、これだけでは「3.」（p.1）の答えにはなっていませんのでそれらを含めて、詳しいの実装の解説をしてみたいと思います。

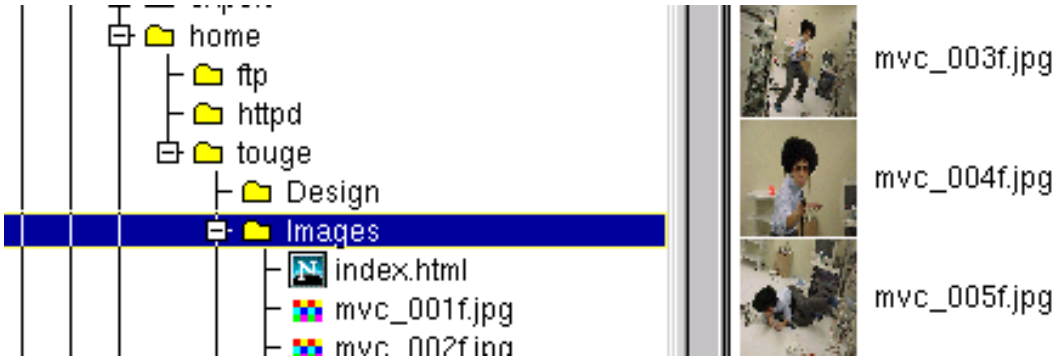


図 1: ローカルファイルシステム

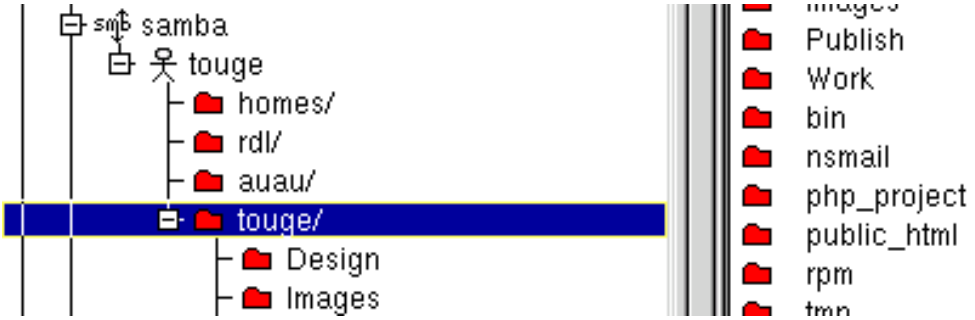


図 2: samba サーバへのアクセス

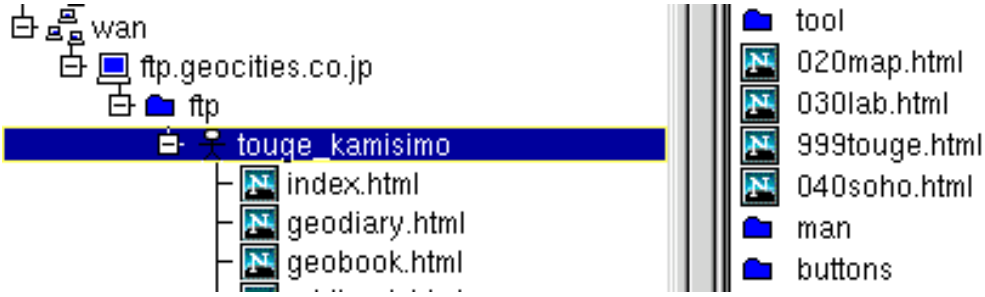


図 3: ftp サーバへのアクセス

2.2 システム要件

GTK が使える環境であれば、大概のディストリビューションで問題ないと思いますが、現在の開発環境は表 1 のようになります。非常にシンプルなコンセプトのものなので、Qt などの他のツールキットへも比較的楽に移植できるのではないかと思っています。尚、ツールの名称は（仮に）「Tree Shell」としています。

名称	Tree Shell(仮称)
CPU/MEM	Celeron 350MHz/128 MB
OS(Distribution)	Vine Linux 2.1.5
LANG/GUI Lib	C++/GTK 1.2.8

表 1: 開発環境

2.3 汎用性を求めて

図 1〜図 3（p.2）では、ローカルファイルシステム/SMB/FTP へのアクセス機能が紹介されましたが、そのような特定のアプリケーション（またはプロトコル）のためのツールではなく、もう少し汎用的な目的に耐え得るような実装を意識しています。

2.4 実装概要

（汎用性追求における）実装面の特徴は、図 4 のようになります。システム的に大きく二つのパートから成っており、一つが C++ で書かれたエンジン部分で、描画とコールバック時のダイナミックな型（アプリケーション）判定などを受け持ちます。もう一つが、外部的に定義されたアプリケーション部分で、ftp や smb などの実際の挙動に関する記述はこちらで行います。

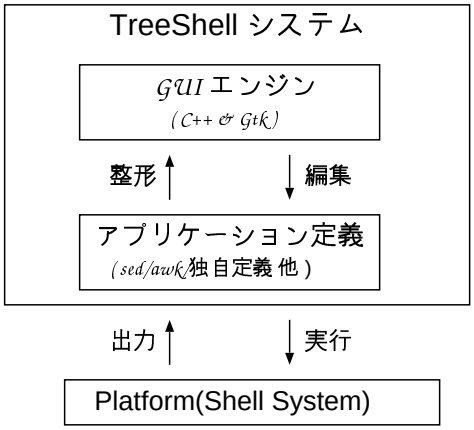


図 4: Tree Shell の実装構造

2.5 利便性を求めて

名称を Tree “Shell” としていますが、「sh」や「csh」などを多少意識しています。とはいうものの、GUI ベースのツールですから、簡単なマウス操作でいろいろなデータを扱えるのは勿論ですが、特に、キャラクタベースの Shell では扱いづらい画像データなどを簡単に扱えるように工夫しています。(図 5 参照)

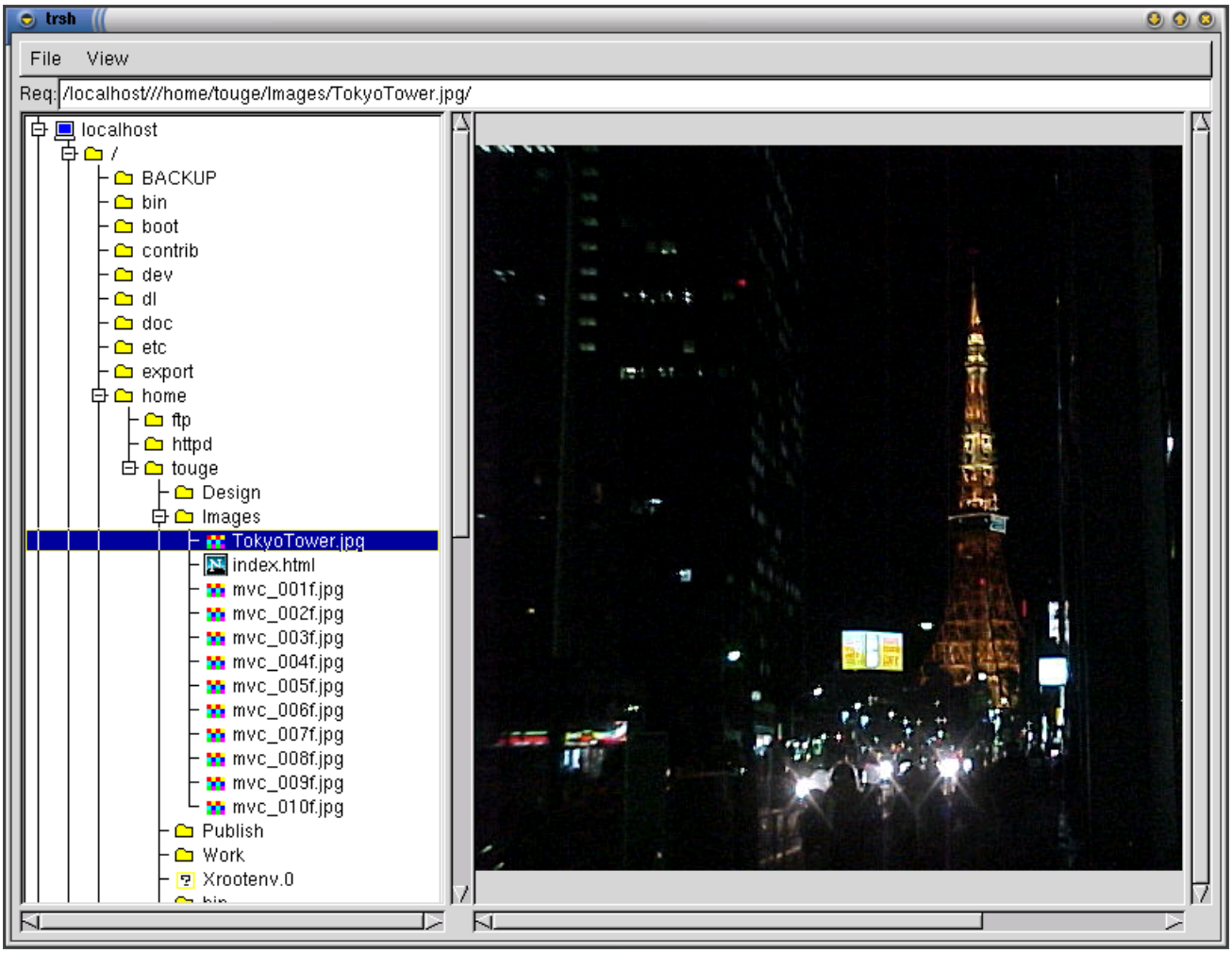
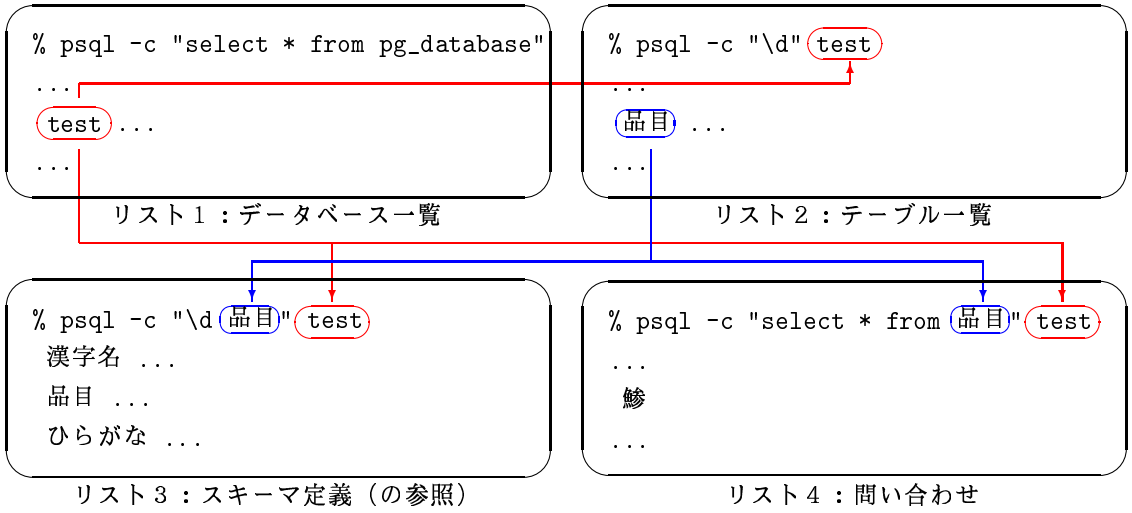


図 5: JPEG 画像の表示

3　ソフトウェアのパターンと再利用

3.1　情報の連鎖とツリー構造

ファイルシステムを管理するためのツリー構造の利用はよく見かけます。そのほとんどは、i-node の構造に対するアプローチなのではないかと思いますが、Tree Shell の場合は、そのように特定のデータ構造に対するものではなく、「人によるアプリケーションやデータに対する処理の一連の連鎖」とでもいったものに対してアプローチする事を前提としています。RDB を例にとってみましょう。具体的に PostgreSQL のコマンドの連鎖をリスト 1 〜リスト 4 に示します。あるコマンドの出力を次のコマンドのパラメータに利用している事が分かります。Tree Shell では、このようなコマンドの出力結果を再利用することによって、マウス操作によるコマンドの連鎖を実現しています。（図 6 参照）



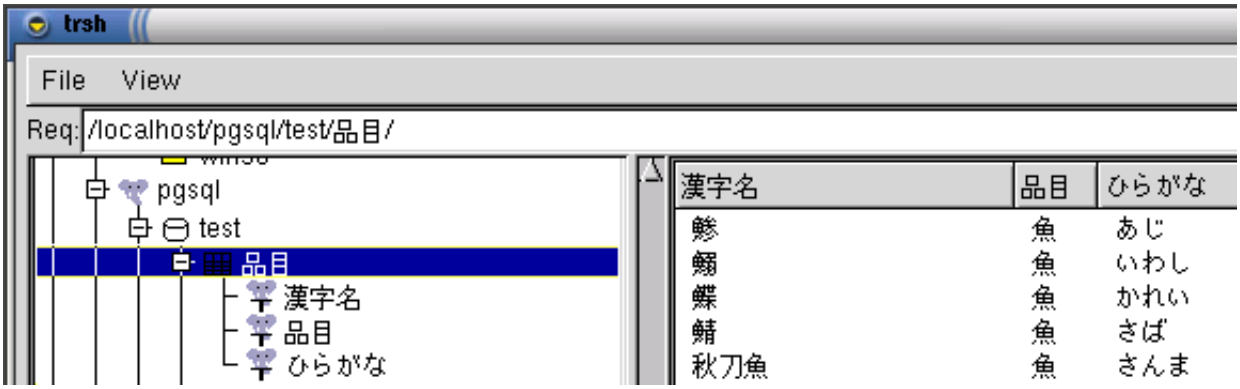


図 6: リスト 1 〜 4 の処理を Tree Shell 上で実行

3.2 パターン分析 1 - ユーザの行動 -

先の RDB の例では、ある特定の（あらかじめ定められた）パラメータを使ってアプリケーションからデータを抜き出す（参照する）という特殊なパターンでしたが、より一般的なユーザ行動として、データを更新するための処理や任意のパラメータを利用したコマンド実行などを考慮にいれるべきでしょう。妥当かどうかは分かりませんが、適当に類型化して例示を試みたものが表 2 になります。

処理パターン / パラメータ	更新無	更新有
定型（または再帰的）	ファイルシステム参照 (ls など) 標準出力 (cat, more など)	発番／ローテーション (バックアップやログ)
任意	検索系 (grep など) 認証系 (ftp, smb など)	ファイルシステム更新 (touch, mkdir など) アカウント更新 (useradd など)

表 2: ユーザ行動のパターン

（表 2 の下部の）ユーザ認証やアカウント作成などの行為は（どんなに工夫しても）ワンクリックで実現する事は難しいことのように思えます。よって、「任意の（値を必要とする）パラメータの実行に際しては、キーボート入力を受け付ける GUI ウィジェット (部品) が必要」という認識を得ることになります。（というか、勝手にそう思ってしまいました。）

少し強引な展開のような気もしますが、ともかくも、HTML の form タグのような機能があるとツールとしてもいろいろと広がりが期待できる事は容易に想像出来ます。で、実際に実装したイメージが図 7 になります。

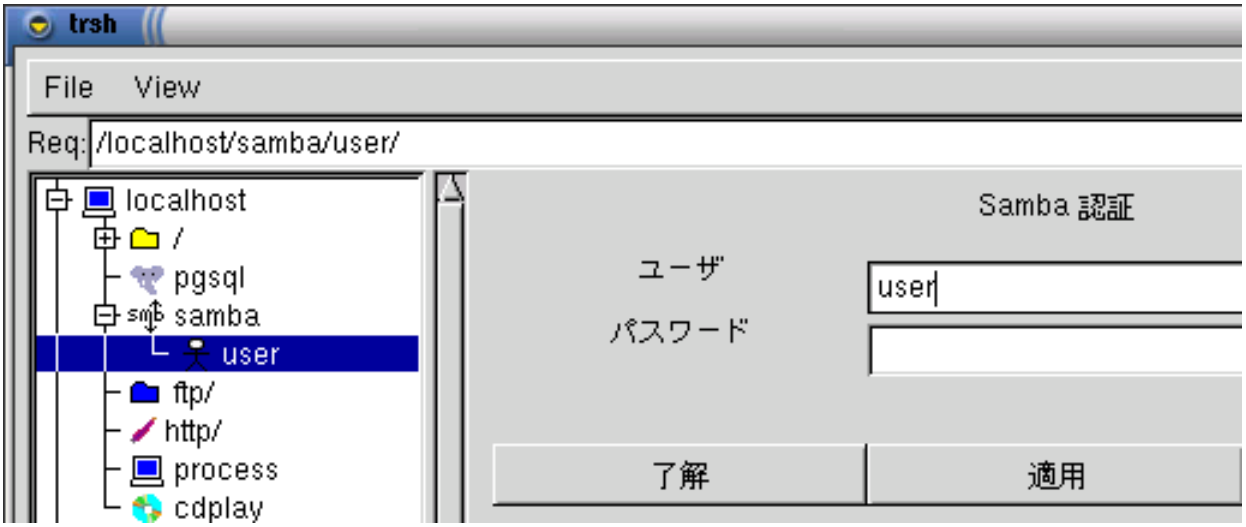


図 7: 任意パラメータコマンドの実行 (Samba 認証)

3.3 パターン分析 2 - アプリケーションの構造 -

前節でのように、ユーザ（にありがちな）行動をある程度考慮してみることで、必要とされるインタフェースの輪郭がおぼろげならも見えてくるかと思います。かなり大雑把ではありますが、現時点での Tree Shell の GUI に関する構造を示したものが図 8 になります。これはあくまで分析レベルという事でかなりデフォルメしたモデルになっていますが、総合的・論理的にデータをハンドリングするベースとして「ツリー」クラスが中心的な役割を果たし、よりディテールを強調したり、より人間の直感にうったえる形式を提供するのが「キャンバス」クラスということでバランスをとりたいので、このような構図（パターン）を念頭に置いておくのは効果的のようです。（何も意識しないでいると、局所的なコーディングに気を取られて、この原則を破ってしまって結果的に持続的な改良が困難になってしまってなかなか大変な思いをします（というか、実際にしました）。）

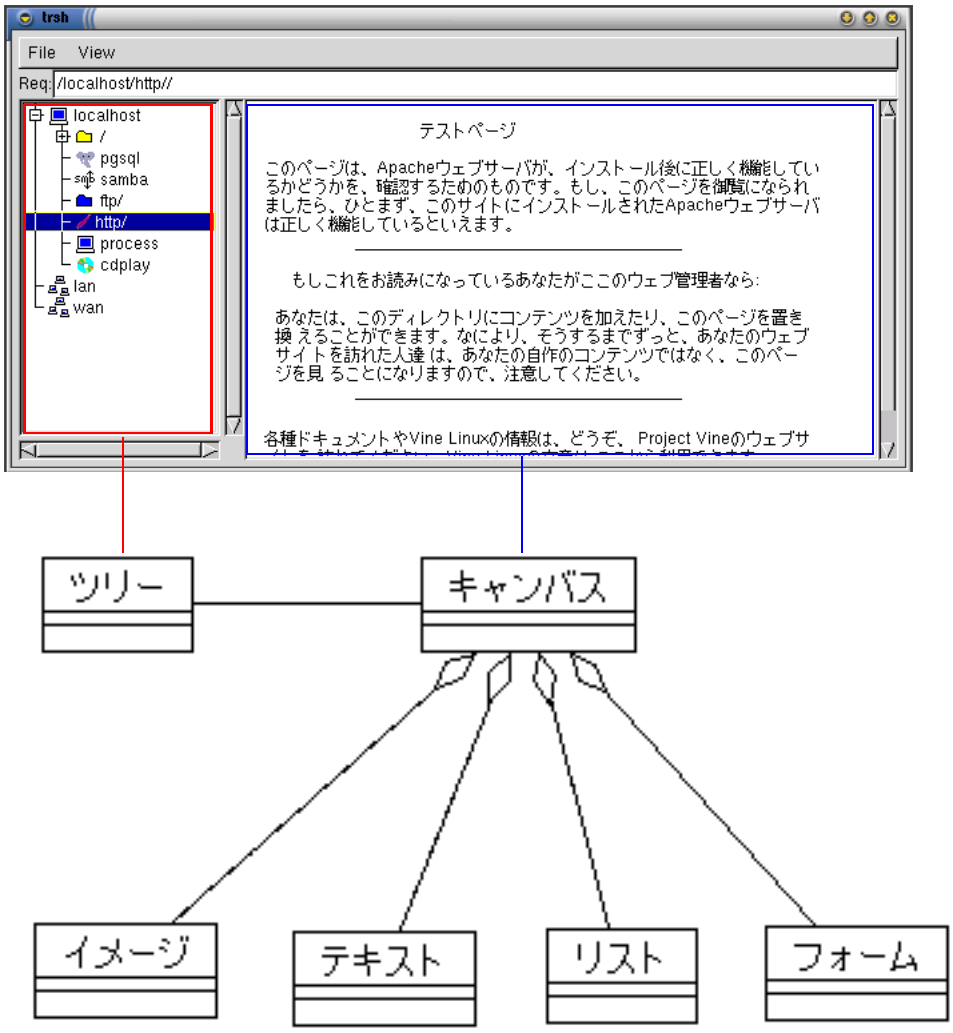


図 8: アプリケーション構造（クラス図）

3.4　2 つの再利用

このように、人のよくある行動パターンをつぶさに眺めて (コマンドの連鎖のパターン分析)、それに対して類型化しつつもある程度の汎用的なレベルを保った器 (簡単な GUI 部品) を与える事が出来ました。これによって何が出来るようになったのでしょう。実際にツールのプログラムを作成しているとより実感することなのですが、このツール (Tree Shell) 自体は以下 2 つの意味での再利用メカニズムを実装している事になります。

- コマンドラインベースのソフトウェア全般
- 上記ソフトウェアの出力結果

ちなみに、前者はあらかじめユーザサイドでどのコマンドを再利用するかを定義しておく必要があります。後者の方はツールの実行時に決定されていくのでよりダイナミックなものと言えますし、前者の再利用を補完するための補助的なものと言えなくもないでしょう。ともかくも、このような形で当初の前提「3.」(p.1) を実現している事になります。

3.5　余談「パターン」

「パターン」という用語を多用してしまいましたが、ここでいうパターンはいわゆる「デザインパターン」[1] などとして一部に定着しているものとは少し違った意味合いになってるかと思います。ただ、その概念の出発点になっている「パターン」[2, 3][1]に近いものではないかなあと思っています。

(オリジナルである建築の世界での話にしても、ソフトウェアでの議論にしても) あるフォーマットに従った「パターン」の辞書を作って活用しなさい、というような事がよく言われているようですが、ともかくも過去の知識 (成功例) を丹念に積み重ねる事とそれを上手に利用するという行為 (仕組みと言うべきでしょうか) の重要性を説いているかと思います。そういった文脈に適当に当てはめてみると、Tree Shell システム (p.3-図 4) のエンジン部分はパターン辞書のフォーマットであり、アプリケーション定義部分は実際の辞書の中身という事にでもなるのかなあと思いますが、どうなんでしょうか。

[1]特に、[2] に関しては難解な内容でもあり、解釈の自由度は高いと思いますので。

4　おわりに

4.1　シンプルに

エンジン部分のコードは極力シンプルなコンセプトにするよう、特定用途向けの処理は出来る限り排除するように心がけていますが、いろいろな意味で難しいなあと思います。そのかいあってか、C++ ベースのコードとしては、たかだか 3000 ライン程度に収まっていますので、適応範囲の広さのわりには実行サイズもコンパクトで気軽に利用できるのではないかと思います。実用レベルに持っていくためにはもう少し作り込まなくてはならいとしても、どのみち数千ラインのオーダーをこえることはないでしょう。

4.2　マウスとキーボード

個人的に、「GUI はマウス、コマンドラインはキーボード」という認識がありましたので、このようなツール (GUI ベースのツール) の場合、どんなに頑張っても既存のコマンドラインユーザに訴求するものを作成することは不可能だろうと思っていました。しかし、(Tree Shell 上で) キーストロークのみで日常的な事は済ませる工夫は出来ないものかという思いが断続的に湧いてきてしまいます。実際、Windows や Mac のユーザでもショートカットキーベースでかなりの事をこなす人々を見かけることがありますし、何とかしたいものです。

4.3　アプローチの是非

実装プログラムの紹介が中心の内容になりましたが、まずは、このようなアプローチ（実装自体ではなく実装方法とか方針といったレベルのもの）に可能性があるかどうかが分かりませんので、いろいろな角度から検証していければと思っています。（今回の発表もその活動の一つです）

4.4　ブラックボックス利用とライセンス

実装アプローチと関連して、最近は、オープンソースと知的財産権[2]の問題に注目が集まっているかと思いますが、このようなブラックボックスでのソフトウェア再利用を前提としたツールの場合、ライセンス的にもより柔軟な展開が可能ではないかと思いますが、なかなか重い話題なので慎重な議論が必要かとは思います。

4.5　オープンに

最後になりましたが、このツールは「オープンソース」[3]を目指しており、現在もそれに向けて作業中です。ついでといっては何ですが、クラス図などの設計情報などもオープンにしようと思ってます[4]。

[2]「特許権・実用新案権・意匠権・商標権の工業所有権と著作権の総称」と辞書にありました。
[3]ソースをオープンにしたからといって「オープンソース」と呼べるわけではないようなので、何をすればいいのか勉強中です。
[4]クラス図をオープンにしたからといって「オープンデザイン」と呼べるわけではないと思いますが。そもそもそんな用語は聞いた事ないですね。

A　ダウンロード情報

本稿に関連したリソースに関しては、以下のものがダウンロード可能です。

ソース ……http://www.geocities.co.jp/SiliconValley-Cupertino/8538/trsh/TreeShell-preAlpha.tgz

クラス図 1 …http://www.geocities.co.jp/SiliconValley-Cupertino/8538/trsh/model.jpg

クラス図 2 …http://www.geocities.co.jp/SiliconValley-Cupertino/8538/trsh/design.jpg

参考文献

[1] Erich Gamma,Richard Helm,Ralph Johnson,John Vlissides *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995

[2] Christopher Alexander, *Timeless Way of Building*, Oxford Univ Pr, 1979

[3] Christopher Alexander,Sara Ishikawa,Murray Silverstein, *A Pattern Language: Towns, Buildings, Construction*, Oxford Univ Pr, 1977