社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－DD－47 (1)
2004／11／26

複数の視点から情報を管理するファイルブラウザの開発

八重樫 尚彦† 白井 靖人‡

一般的なOSのファイルシステムはツリー構造を採用し、ファイルの階層的な分類、管理を支援している。これは、人間が情報管理をする際に分かりやすいという利点があるが、構造を一つの視点に固定してしまうため、物事を分類することに関して脆弱な部分がある。本来、情報は異なる視点で分類、利用することが可能である。そこで、ユーザの利用目的に応じた、複数の視点による情報管理が可能なファイルブラウザを開発する。このファイルブラウザは、既存のファイルシステムのツリー構造に加えて、利用者が異なる視点で構築した仮想的なツリー構造を用いてファイル管理をするというものである。

A File Browser with Multiple Views on Information Management

Yoshihiko Yaegashi† Yasuto Shirai‡

The file system of general OS adopts a tree structure, and is supporting the hierarchical classification of a file, and management. Although there is an advantage that it is intelligible for man when managing information, there is a brittle part about classifying since structure is fixed to one view. Originally it is possible to classify information according to multiple views and to use. So, we develop a file browser with multiple views on information management. This file browser manages a file using the virtual tree structure built with multiple views in addition to the tree structure of the existing file system.

1. はじめに

現在、パソコンを代表とする情報処理装置は、高性能化、小型化、低価格化が進み、企業や家庭へ急速に普及している。このように情報化が進むにつれて、各種文書、画像、音楽といったデジタルコンテンツが増え、情報の利用者や利用目的が多様化している。そのため、膨大な数のファイルの管理が重要となってきている。

一般的なOS(Windows、UNIXなど)のファイルシステムはツリー構造を採用し、ファイルの階層的な分類、管理を支援している。これは、人間が情報管理をする際に分かりやすいという利点があるが、構造を一つの視点に固定してしまうため、物事を分類することに関して脆弱な部分がある。本来、情報は異なる視点で分類、利用することが可能である。そこで、ユーザの利用目的に応じて、複数の視点による情報管理が可能なファイルブラウザを開発する。

開発するファイルブラウザは、既存のファイルシステムのツリー構造に加えて、利用者が異なる視点で構築した仮想的なツリー構造を用いてファイル管理するというものである。利用者は複数のツリー構造を操作しているように感じるが、実際はそれらのツリー構造で一つのネットワーク構造を形成している。これにより柔軟な情報管理が可能なGUIシステムとして機能する。本研究のファイルブラウザは、現在のファイルシステムに欠けていた点を補完するものとして位置付けることができる。

2. 現在のファイルシステム

ユーザがパソコンで処理するファイルやディレクトリ(フォルダ)を、実際の物理的なディスク上に対応させ、ファイルやディレクトリ自体を構成する体系をファイルシステムという[1]。

ファイルシステムの多くはツリー構造を採用している。その構造は一番上に根(root)があり、ディ

†静岡大学大学院情報学研究科
Graduate School of Information, Shizuoka University
‡静岡大学情報学部情報科学科
Faculty of Information, Shizuoka University

レクトリを節(node)、ファイルを葉(leaf)とする。そして、つながっている節と節で、根に近いものを親、そうでないものを子という。

Windowsではディスクドライブごとにディレクトリツリーを作って管理しているため、“A:¥”、“C:¥”などのように、ドライブ文字のみで表されているディレクトリがそれぞれのディスクのルートディレクトリになる。ただし、ハードディスクを複数のパーティションに分割して使う場合は、パーティションごとにルートディレクトリができる。

UNIXでは全てのディスクを一つのディレクトリツリーの下に置いて管理する方式をとっているため、“/”というディレクトリ一つだけがルートディレクトリになる。

ツリー構造の中で、個々のファイルやディレクトリを指すための表記が、ファイルパスである。ツリー構造の中でそのファイルまでをたどっていく道順を示している。パスがあることにより、同じ名前のファイルがあったときに区別がつくようになる。パスを使ったファイルの指定には絶対パスと相対パスの二つの方法がある。

絶対パスは、指定したいファイルを最上位の項目から順番にたどりすべて記述して指定する方法である。絶対パスでファイルを指定する場合は、ルートディレクトリを基点とし、道のりとして通るディレクトリを順に、区切り文字で区切って表す。Windowsでの区切り文字は“¥”、UNIXでは“/”となる。例えば、

/home/program/report.java

というパスは、report.javaファイルまで、ルート、home、programというディレクトリを通る。

相対パスは、指定したいファイルをカレントディレクトリから見た位置として指定する方法である。相対パスでは、起点となるディレクトリを“.”で、上位ディレクトリを“..”で表す。例えば、

../../test.txt

というパスは、現在のディレクトリの二階層上位にあるディレクトリの中にあるtest.txtというファイルを表す。

現在のファイルシステムでは、一つの視点から構築されたツリー構造を用いてファイル管理を行っている。そのため、一般的なファイルブラウザは一つのツリー構造を忠実に表示したものでファイル管理を行うことになる。

このようなファイルブラウザでは、ファイル管理時に、“新しいツリー構造にしたい”、“前のツリー構造に戻したい”といったような異なる視点からなるツリー構造を利用したい場合には、ツリー構造を再構築しなければならない。既存のツリー構造が巨大で、分類が乱雑であればあるほど、ツリー構造の再構築は利用者に負担を掛けることになる。

本研究では、同時に複数のツリー構造を用いてファイル管理を行うファイルブラウザを開発する。これにより一般的なファイルブラウザでは不可能であった、複数の視点からのツリー構造を構築することができる。

3. ファイルの整理法

3.1 分類するということ

ファイルを整理し組織化するための最も基本的な手段は、分類することであり、その基本は、同じようなものをまとめ、異なるものを分けることである。ツリー構造と分類のおかげで、探しているファイルを比較的簡単に見つけることができる。しかし、この“分類”は、人間の認識や理解といったものと密接に関連しており、かなり複雑な問題が存在している[2]。

分類の難しさの多くの部分は、分類体系の構築、すなわち、分類カテゴリ集合の設定と体系化に関わった問題である。分類が効果的に働くかどうかは、適切な分類体系を設定できるかどうかにかかっている。それでは、ファイルを分類するときに、どのような分類体系を設定するのが適切なのだろうか。これは大変難しい問題であるが、“分類がファイルの利用目的に合っていること”、“検索が容易なこと”はファイル整理において重要なことである。

分類には必ず目的がある。例えば、音楽ファイルをアーティスト別やジャンル別に分類する。これは、ファイルの管理と使用時の利便性を向上させることを目的とした分類である。また、この目的には検索を容易にするという意味もある。このように、それぞれの分類にはそれぞれの目的があり、それぞれの目的に合った分類体系が必要である。もし分類体系が目的に合っていなければ、その分類は何の意味も持たない。また当然のことながら、分類対象が同じでも、異なった分類体系が必要となる場合も十分に考えられる。

ファイル整理において、検索の容易さに重点をおいた場合、その分類体系が検索に有効に働かなければならない。そのためには、入手したいファイルの分類カテゴリが容易に推定でき、分類カテゴリを特定することによって、事物の数が十分に絞り込まれるということが重要である。これはフ

ファイル数が膨大になればなるほどである。

ファイルを分類するときや分類後の管理のときに、必ずと言っていいほど問題が起きる。それは4節で示すこととする。

以下に具体的なファイル整理の方針を示していく[3]。

3.2 名前優先

単純に整理するものに付けられた名称の50音順やアルファベット順で分類する。日本語には漢字、カタカナ、ひらがな、アルファベット表記が混在している。コンピュータの名前順によるソートはあくまで文字コード順であることを考えなければならない。商品名や人名などのように、膨大な数のデータを扱う可能性がある場合や、普遍的なファイル名が存在する場合に有効である(図1)。

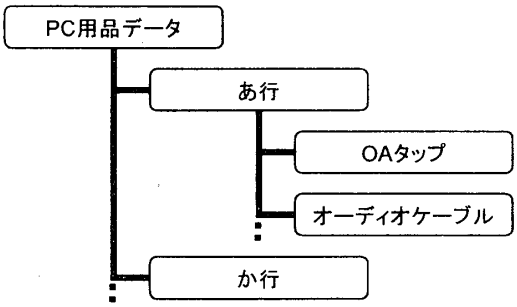

図1：名前優先のツリー構造

3.3 時間優先

時間単位で一区切りするように考える。ある一定時間に関係するものという単位で、ある程度の機械的な分類をすることができる。現在関わっている業務で、頻繁に使用するファイルにアクセスしやすい。例えば、年度単位のディレクトリを作り、その中にファイル内容ごとのサブディレクトリを作成するという方法が考えられる(図2)。

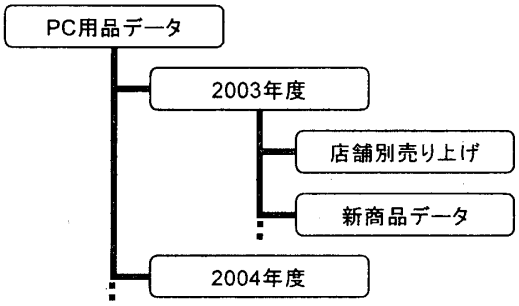

図2：時間優先のツリー構造

3.4 内容優先

ファイルの内容によって区別する。厳密に分類しようとすればするほど問題が大きくなる。利用者の目的に合う分類方法を考えることが必要となる。継続的に行う業務が多い場合は、まず業務ごとのディレクトリを作り、その中に年度別のディレクトリを作る方法が考えられる(図3)。

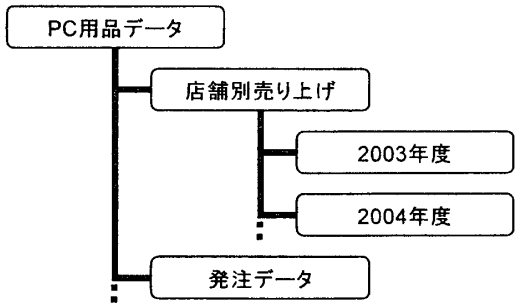

図3：内容優先のツリー構造

4. ファイル管理におけるツリー構造の利点、問題点

3節で、ファイル整理時における分類の重要性と具体的なファイル整理の方針を示した。この節では、現在のファイルシステムにおいてツリー構造でファイルを管理することの利点、問題点を記述していく[4]、[5]、[6]。

4.1 利点

ツリー構造でファイルを管理することの利点は以下のようなものがある。

(1)複数のファイルをまとめることができる。

簡単に同じ目的で集められたファイル集合を定義することができる。それによってファイルの使用と管理を手助けすることができる。

(2)検索が容易である。

3節でも述べたように、入手したいファイルが存在する分類カテゴリを推定するということが、ファイル検索を手助けしている。入手したいファイルの分類カテゴリを特定することによって、事物の数を十分に絞り込むことができる。

このような観点から、類似する他のファイルを探す場合にも、ツリー構造は有効であることが分かる。

(3)関係性の表明、強調ができる。

ツリー構造によってノード間の関係が明らかになる。データとデータの関係に関連性をつけて区

分し、データとデータの関係を階層として図式的に捉えるということである。例えば、全体と部分、原因と結果、過程と成果物など、様々な切り口で関係性を示すことができる。そして、3節のファイル整理の例のように、上位ノードに強調したいデータを配置することができる。

(4)空間的、機能的な距離を表現できる。

分類される事象同士の関係の程度を示すことができる。例えば、大学と学部のような縦の関係や、同じ大学の教育学部と工学部のような横の関係を示すことができるということである。

4.2 問題点

ツリー構造でファイルを管理することの問題点には、様々なものがある。以下にそれを示す。

(1)ファイルはその内容に多面性を持つ。

ファイルは複数のディレクトリに所属できる場合がある。図4のような有名な例がある。“こうもり”は“飛ぶ”という分類カテゴリと“哺乳類”という分類カテゴリのどちらにも属することができる。現在のファイルシステムではルートノードから目的のファイルにたどりつく道は、ただ一つしか許されていない。そのため図4のような場合、どちらか一方の性質を無視して考えるしかない。

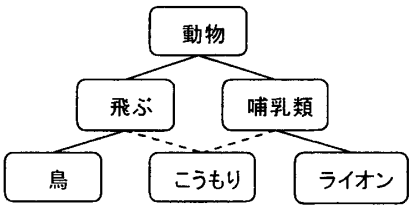

図4：データ項目の多面性

(2)情報は時間経過により変化、誕生、消滅する。

情報というものは単純ではなく、利用するときの状況、目的等により変化する可能性がある。また、時間経過により様々な概念が誕生し、消滅する。そのため、ある時点で最も美しく整理できたと思った分類が、しばらくすると破綻するということが考えられる。

(3)利用者の目的により分類構造が異なる。

一つのツリー構造でしか表せないため、目的によっては使いづらいファイル、ディレクトリの構造となってしまう。この問題は利用者が一人なのか複数なのかということにも関係する。利用者が増えれば、その分、利用目的も増えるだろう。

(1)～(3)の問題点の対応策として、複数のツリー構造を用いてファイル管理を行うということが考えられる。現在のファイルシステムに管理を頼る限りでは、一つのツリー構造で情報を管理し続けなければならない。これはある一つの視点でのみ成立している構造である。情報は一つの構造でまとめられるほど単純ではないのにそうしなければならない。情報の分類対象が同じでも、異なった分類体系が必要となる。

そこで、本研究では、ファイル管理におけるツリー構造の問題を解決するために、複数のツリー構造を用いて、複数の視点からファイルを管理できるファイルブラウザを開発する。5節で、そのファイルブラウザの詳細な説明を示していく。

5. ファイルブラウザの開発

5.1 ファイルブラウザの概要

開発したファイルブラウザは、既存のファイルシステムのツリー構造に加えて、利用者が異なる視点で構築した仮想的なツリー構造を用いてファイルを管理することを可能にする。これは4節で示した現在のファイルシステムの問題点を解決するシステムとなっている。

仮想的なツリー構造を構築するには、まず仮想的な親ノードを作成し、既存のファイルやディレクトリをその子ノードとして登録する。こうすることによって、子ノードは複数の親ノードを持つことになる。全体として見ると、それぞれの視点が独自の親を設けるのを許すことによって、利用目的に応じた複数のツリー構造を構築することになる。

仮想的にではあるが、複数の親を持たせた構造は、ツリー構造ではなくネットワーク構造である。しかし本研究のファイルブラウザでは、そのネットワーク構造を分解し、複数のツリー構造として表示している。そのため、ユーザはネットワーク構造を感じることなく、複数のツリー構造としてファイル管理をすることができる。

図5、6、7に具体的なイメージを示す。まず、既存のツリー構造(図5)の“写真”と“img”に“画像管理”という仮想的な親を作成し、それぞれを子ノードとして登録する。こうして作成したツリー構造を図6に示す。このツリー構造には、画像データを管理するという目的がある。

本質的には、ノードに複数の親があると、図7のようにネットワーク構造になってしまう。今回

開発したブラウザの画面では図 5 のツリーと図 6 のツリーを切り替えて表示するため、感覚的には複数のツリー構造としてファイルを管理しているようになる。

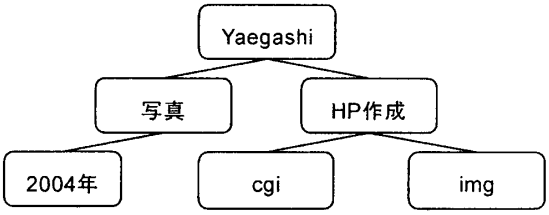

図 5：既存のツリー構造

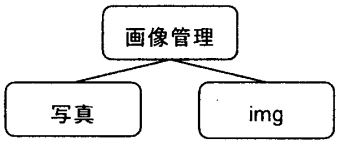

図 6：仮想的なツリー構造

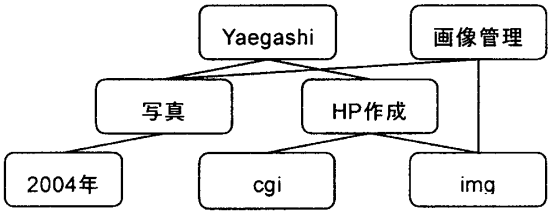

図 7：図 5 と図 6 の本質的な構造

本研究のファイルブラウザの使用時の流れは次のようになる。より具体的な操作方法や入出力例を 5.2～4 節に示す。

(1)仮想的な親ノードを作成し、その子ノードに既存のツリー構造のファイル、ディレクトリを登録する。
(2)各種設定をする。
(3)設定にしたがって、新しいツリー構造を作成する。
(4)構築した複数のツリー構造でファイルを管理する。

5.2 ファイルブラウザの画面構成

開発したファイルブラウザの画面例を図 8 に示す。タブを用いてツリーを切り替えるという部分が、一般的なファイルブラウザと大きく異なる[7]、[8]。このタブ名には、ツリーを構築する際の視点に当たるものを定義することができる。3.1 節でも述べたように、分類には必ず目的がある。タブ名にその目的を示す名称を付けることにより、ユーザによるツリー構造の管理を支援することができる。

このファイルブラウザの初期状態で、“original”という視点名のツリーがある。これは OS のファイルシステムのツリー構造である。この視点名は固定となっている。

また、開発言語には Java を用いている。Java2 SE v1.4 以降のバージョンで動作する。

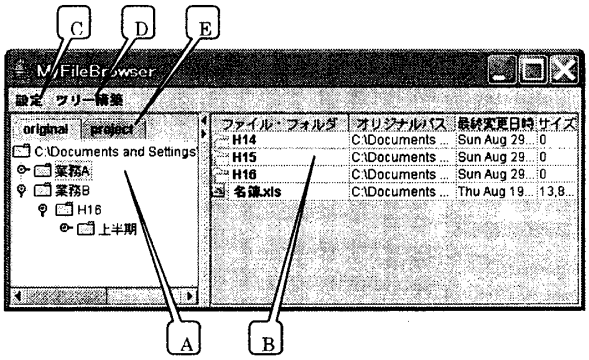

図 8：ファイルブラウザの画面例

以下に図 8 の説明を示す。

A：ファイル、ディレクトリのツリー構造を表示する領域
仮想的なツリー構造を表示している場合、右クリックにより仮想的な親とその子ノードの登録、抹消を行うことができる。

B：ファイル、ディレクトリのリストを表示する領域
ファイル、ディレクトリ名、既存のツリー構造でのパス名、最終変更日時、サイズが表示される。また右クリックにより、仮想的な親を作成し、その子ノードに、選択したファイルまたはディレクトリを登録することができる。

C：各種設定を行うメニュー
“半自動的に仮想的な親の子ノードに登録”、“ファイルのみをノードに登録”の設定をすることができる。チェックボックスにチェックを入れることにより、その設定が有効となる。

D：ツリー構造を作成するメニュー
新しく仮想的なツリー構造を作成し、タブ(E の部分)に追加する。

E：ツリー構造の表示を切り替えるタブ

タブを切り替えて、領域 A に表示するツリー構造を選択する。

5.3　ファイルブラウザの機能

このファイルブラウザで実現した機能を示す。

5.3.1　仮想的な親の作成と子ノードの登録

右クリックにより、仮想的な親を作成し、その子ノードに、図 8 の領域 B で選択したファイルまたはディレクトリを登録することができる(図 9)。一度作成した仮想的な親はリストから選択することができる。

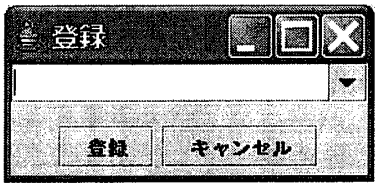

図 9：登録画面

5.3.2　半自動的に仮想的な親の子ノードに登録

仮想的な親の作成前に、必要に応じて“半自動的に仮想的な親の子ノードに登録”のチェックボックスにチェックを入れる。これにより、既存のツリー構造の中にある、仮想的な親の子ノードとして登録するファイル、ディレクトリと同名のもの全てを、半自動的に登録することができる。

この機能は同じ名前のファイル、ディレクトリをいくつも登録する仕事量を軽減させるためのものである。

例を図 10 に示す。このツリーは授業に関するウェブサイトのデータを管理しているものである。異なる視点でファイル管理を行うために、“画像管理”という仮想的な親を作成し、その子ノードに“img”を登録する。しかし、このツリーには“img”が二つあるため、二度の登録作業が必要となる。そこで、“半自動的に仮想的な親の子ノードに登録”を用いると、どちらか一方の“img”の登録により、もう一方の“img”も登録され、作業が一度で済む。

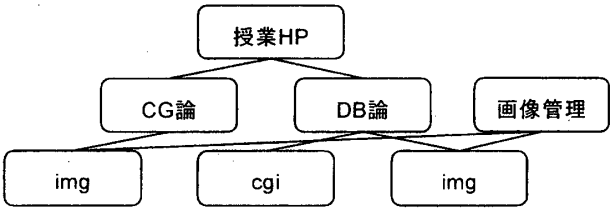

図 10：半自動的に仮想的な親の子ノードに登録

5.3.3　仮想的なツリーの構築

本研究のファイルブラウザは、作成した仮想的な親とその子ノード(既存のツリーに存在するファイル、ディレクトリ)を用いて、ファイルシステムのツリーとは別の、新しく仮想的なツリーを構築することができる。ツリーを構築する前に、そのツリーの視点名を入力する(図 11)。また、仮想的なツリーのルートノードには“root”があり、そこにノードを登録していくことになる。

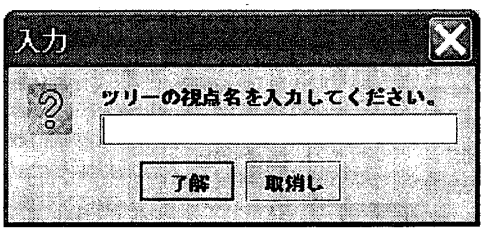

図 11：視点名の入力

図 8 の領域 A で右クリックすることにより、選択されているノード(ディレクトリのみ)に登録、抹消処理を行うことができる(図 12、13)。実際に登録することができるのは、作成した仮想的な親である。登録時には、仮想的な親の子ノードも同時に登録される。

仮想的なツリーには複数の仮想的な親を登録することができる。これにより既存のファイルシステムのツリー構造に近い、柔軟なツリー構造を作成できる。

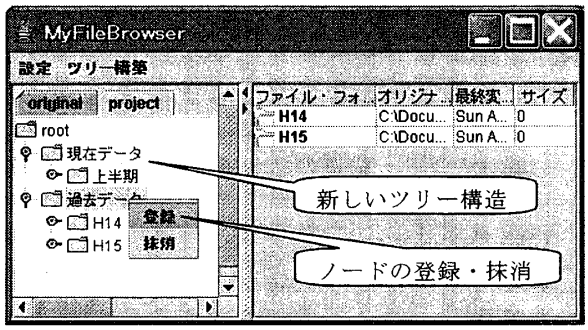

図 12：仮想的なツリーでのノードの登録・抹消

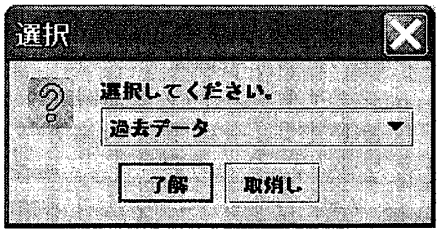

図 13：ノード選択画面

5.3.4　ファイルのみをノードに登録

ツリーを構築する際に、ディレクトリを取り除いて、ファイルのみをノードに登録することができる。この処理をディレクトリではなくファイルに適用した場合には、通常のノード登録処理となる。ファイルを一つの親でグループ化することにより、ツリーのディレクトリ階層が浅くて済むため、目的のファイルにすぐに行き着くことができる。

例を図 14、図 15 に示す。図 14 は、仮想的なツリー構造に、通常のノード登録処理を行ったものである。“過去データ”という仮想的な親を“root”に登録した。“過去データ”のサブディレクトリには“H14”、“H15” ディレクトリが存在する。図 15 は、“ファイルのみをノードに登録”の設定をして、“root”に“過去データ”を登録した。“H14”、“H15” ディレクトリが取り除かれて、末端ノードであるファイルのみが登録されている。

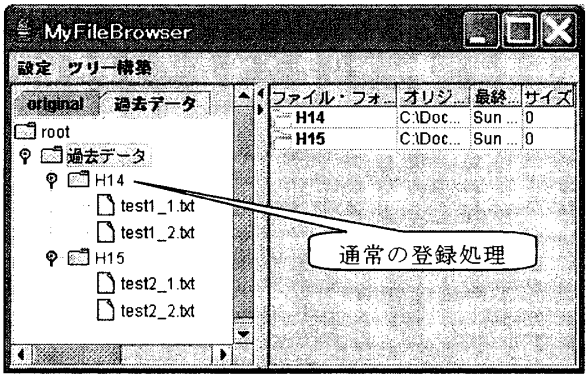

図 14：通常の登録処理を行ったツリー

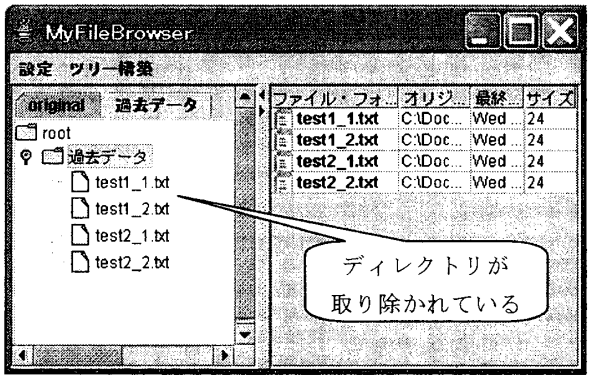

図 15：ファイルのみをノードに登録したツリー

5.4　使用例と結果

開発したファイルブラウザの使用例を示す。

図 16 のような大きく“業務 A”と“業務 B”に分かれているツリー構造を準備する。従来のファイルシステムでは、この構造のみしか表示されない。

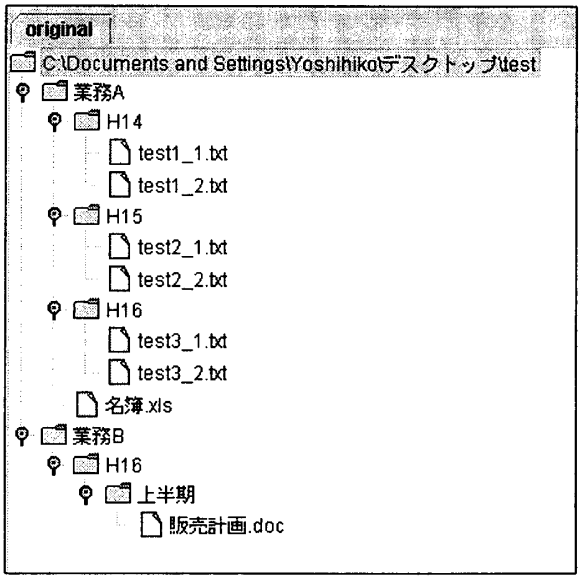

図 16：ツリー構造

このツリーを利用して、新しい業務のための仮想的なツリーを作成する。まず、“上半期” ディレクトリに“現在データ”、“test1_2.txt”ファイルと“H16” ディレクトリに“参考資料”という仮想的な親を作成し、その子ノードに登録する。そして、新しい仮想的なツリーの視点名を“project”として、ツリーを構築する。

その仮想的なツリーの root ノードに“現在データ”を登録した。そして、“ファイルのみをノードに登録”の設定をして、現在データに参考資料を登録した。その結果が図 17 である。

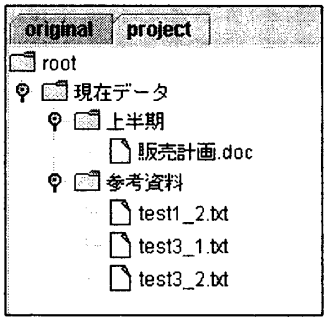

図 17：新しいツリー構造を構築

このように、実際に既存のツリー構造と新しいツリー構造を組み合わせて、ファイルの管理ができることがわかる。

6. 考察と今後の課題

本研究では 4 節の問題点を解決するために、仮想的な親を作成し、新しいツリー構造を構築できるようにした。これによりファイル管理の際に、同時に複数の異なる視点から情報を分類、利用することができるようになった。

しかしこの方法では新たな問題も生じる。複数のツリー構造を構築するための利用者の仕事量が増えるということである。そこで、そういった仕事量を減らす仕組みが必要となる。

本研究では、この問題に対応するために“半自動的に仮想的な親の子ノードに登録”という機能を用意した。しかし、この機能は意図していないファイル、ディレクトリまで仮想的な親の子ノードに登録されることや、同じファイル、ディレクトリ名が少ない場合には効果が薄いといったことが考えられる。

他の問題点として、同じ名前のファイル、ディレクトリの処理が挙げられる。このファイルブラウザの性質上、同じ名前のファイル、ディレクトリが、同じディレクトリに登録される可能性がある。例えば“半自動的に仮想的な親の子ノードに登録”の機能を使用するとそういった問題が起きる。現段階では、既存のツリーのパスと最終変更日時により、同じ名前のファイル、ディレクトリの区別をしている。

今後の課題として、上記の問題点の改善策の検討、通常のファイルブラウザとしての機能の充実の他に以下のようなものが挙げられる

・仮想的なツリーの保存機能の実装
　設定保存ファイルには XML を利用する予定である。

・ファイル抹消の設定の実装
　ファイル抹消の際に、そのファイルのみなのか、他ツリーの同じファイルも全てなのかを選択することができる。

・ツリー構造そのものをコピーする機能の実装
　部分的にツリーの構造を変更したい場合に有効である。新しい仮想的なツリー構造を一から構築する必要がないため、ユーザの仕事量の軽減につながる。

・仮想的な親データを編集するユーザインターフェースの検討
　ツリー構造で表示することにより、GUI で編集を行えるようにする予定である。

7. 参考文献

[1]ITmedia IT 用語辞典、ファイルシステム
http://www.itmedia.co.jp/dict/os/technology/fs/

[2]長尾真 松山隆司 杉本晃宏 佐藤理史 麻生英樹、“岩波講座 マルチメディア情報学 2 情報の組織化”、岩波書店、2000 年

[3]野口悠紀雄、“「超」整理法”、中央公論新社、1993 年

[4]掛下哲郎 原槙稔幸、“多次元分類：木構造分類とキーワード分類の複合的アプローチ”、情報処理学会論文誌（データベース）、Vol.42、No.SIG1(TOD8)、2001 年

[5] 木洩れ陽
http://hp.vector.co.jp/authors/VA027560/index.html

[6] 明示的に宣言されます。
http://www.chimimo.com/mt/

[7]GARY MARSDEN and DAVID E. CAIRNS, “Improving the Usability of the Hierarchical File System”, Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology, pp.122-129

[8]Robert Chin, “Three-Dimensional File System Browser”, Crossroads, Volume 9, Issue 1 (Fall 2002), pp.16-18