システムコールレベルでのバージョン管理機能を持つファイルシステムの研究

01T247　津 紀孝　（最所研究室）

細かい単位でバージョン管理を行うためにシステムコール単位でバージョン管理ができるファイルシステムの研究を行った。バージョン管理ファイルシステムとしてリンク構造のファイルシステムを考え、アプリケーションレベルでの擬似ファイルシステムとして設計を行い、その一部の実装を行った。

1　序章

パソコンの普及によりパソコンで使われるプログラムやデータの量は日々増加している。データは日々変更されているが、変更したことが必ずしもよい結果にならず、変更前の状態に戻したい要求も出てくる。また、現在編集中のデータが破損した場合、過去のデータから元の状態まで戻さなければならなくなる。ファイルの変更後の状態から変更前の状態に戻すには、変更前の状態の管理が必要になる。変更前の状態を人的に管理するコストは、利用するデータの量の増加にあわせて増加する。そのため、ファイルの管理コストを軽減するバージョン管理が必要である。

これまでの本研究室で行ってきたバージョン管理 [1] [2] はブロック単位などのバージョン管理で単位が大きかった。

これに対して、本研究ではシステムコールレベルでの細かいファイルの変化に対応したバージョン管理機能を実現しその有効性を確かめることを目標とした。本研究では実装が簡単なアプリケーションレベルでの擬似ファイルシステムとして実装を行うこととした。

2　概要

write システムコールごとにバージョンを記録するので変更の単位が非常に小さいこともありうる。このため従来のブロック単位でのファイル管理では空間的なオーバヘッドが大きくなる。そこで本研究ではリンク構造を用いることにした。ファイルはリンクをたどることで一つのファイルとなる。リンクに分岐を許すことによりあるバージョンから複数のバージョンを作成することが出来る。

既存のファイルシステム上のファイルをキャラクタデバイスとみなし、そのキャラクタデバイス上にバージョン管理をもつファイルシステムをアプリケーションレベルで実装する。具体的には、バージョンファイルシステムを操作するライブラリを作成する。図 1 に示すように、システムコールの代わりにバージョンを操作するライブラリを使用することでバージョンを管理出来るようになる。このライブラリによってバージョン管理システムを実現する。その特徴は次のとおり。

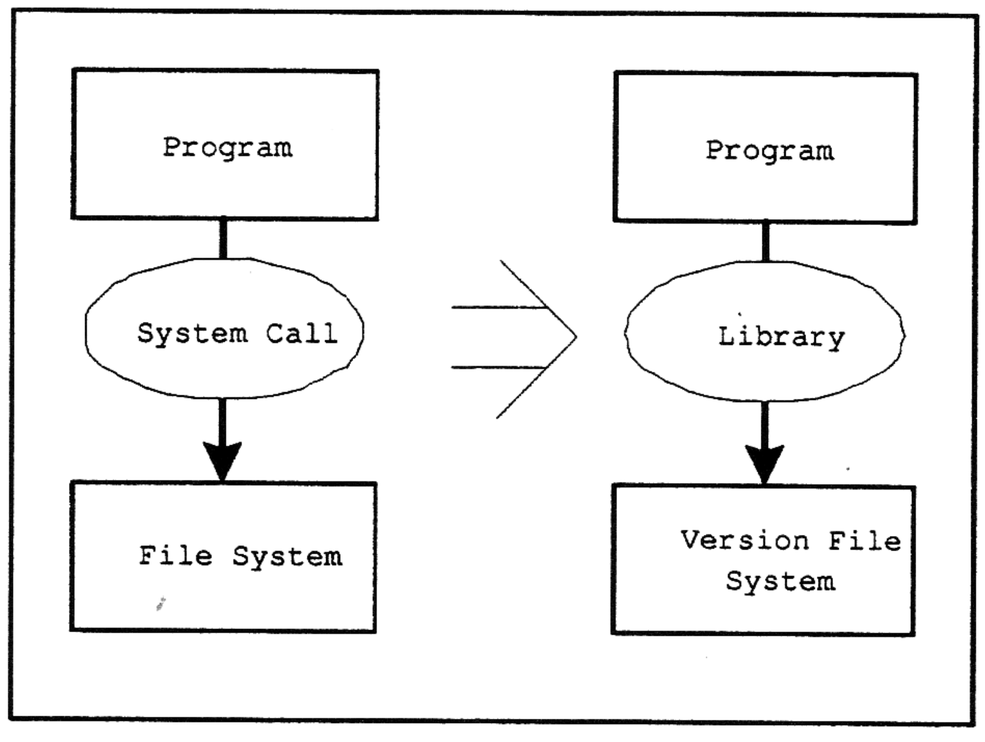

図 1: ライブラリの使用

- OS に組み込むことを想定し、ライブラリの引数などをシステムコールにあわせる。
- ファイルの任意のバージョンで分岐可能である。
- 一回のシステムコール“ write ”が自動的に一つのバージョンとなる。
- ファイルシステムをバイト単位で動作できるキャラクタデバイスとして扱う。
- ファイルは追加のみとする。
- 一回の書き込みを一つのまとまりとして保存し、それらをリンク構造でつなぐ。

ファイルのバージョン管理は、ファイルシステム操作用のライブラリを用いることにより自動的にバージョンを作成し行われる。

3　設計

3.1　ファイル構造

ファイル構造は、以下のようになる。

- スーパブロック – ファイルシステムの大きさとファイルシステムの空の領域へのリンクを持つ。

- ルートブロック – ファイルシステムに存在するファイル名とそのファイルヘッダブロックへのリンクを持つ。
- ファイルヘッダブロック – ファイルのメタ情報を保持し、ファイルデータブロックへのリンクを持つ。
- ファイルデータブロック – バージョン毎のデータを保持する。ファイルデータブロック同士が双方向にリンク構造になっている。このリンクが分岐することで複数のバージョンを作成できる。

これらは、図 2 に示すように互いに関連している。

File Header Block (1) からたどれるバージョンファイルは File Data (1_1) から File Data (1_3) までの一本道であって分岐がない。File Header Block (2) はファイルを作成して中のデータがない状態である。File Header Block (3) からたどれるバージョンファイルは File Data (3_1) から File Data (3_2_1) までのと File Data (3_1) から File Data (3_2_2) の二つある。ファイルオープン時にどちらの分岐を持ったバージョンを開くか指定する。

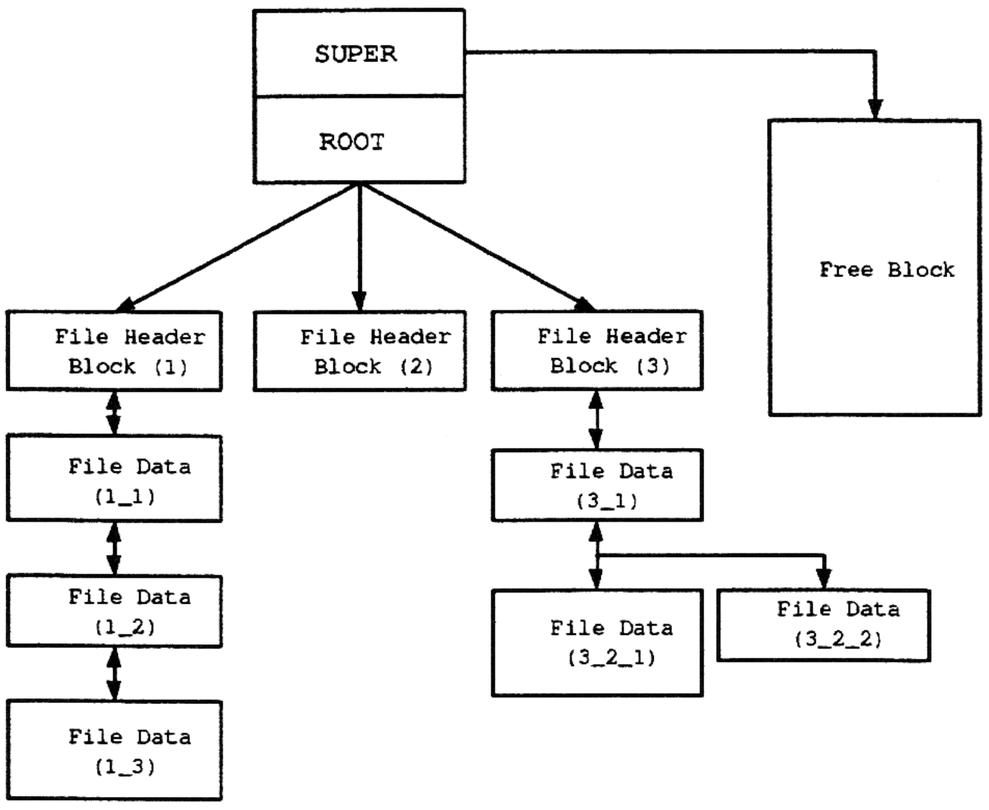

図 2: バージョン管理のデータ構造

また、ファイルシステムとして操作するためのデータ構造として

- ディスクリプタ – バージョンファイルを操作するために必要となる情報を保持する。

を作成した。

3.2 ライブラリ

ライブラリとして以下のものを設計した。

- ファイル作成 – ファイルシステム上に新しいバージョンファイルを作成する。指定されたファイル名をルートブロックに登録し、ファイルヘッダブロックを空き領域から作成する。同一のファイル名を持ったものは作成できない。
- ファイルオープン – 指定されたバージョンファイルを開き、ディスクリプタに割り当て、ファイル操作が出来るようにする。呼び出し時にファイルのバージョンを指定できる。
- ファイルクローズ – ファイルオープンされたバージョンファイルを閉じる。ファイルのメタ情報を書き込む。
- ファイルリード – ファイルオープンで開いたバージョンファイルから指定した分のデータをデータブロックのリンクをたどって読み取る。
- ファイルライト – ファイルオープンで開いたバージョンファイルに空き領域から作成したデータブロックをリンクでつなげることで書き込む。
- ファイルシーク – ファイルオープンで開いたバージョンファイルのディスクリプタが持つ現在位置を指定された分だけデータブロックのリンクをたどって移動させる。

4 実装

3 節の設計に基づいて C 言語でライブラリの実装を行った。ファイル作成、ファイルオープン、ファイルクローズ、ファイルライトについては、実装が終了した。ファイルリードは現在実装を行っているところである。ファイルシークは未実装である。

5 まとめ

本研究ではバージョン管理機能を持つファイルシステムのアプリケーションレベルでの擬似ファイルシステムの設計と一部実装を行った。ライトは実装を簡単にするために追記のみとして実装した。リードは分岐をたどってデータを読み出すところの実装を行っている。今後は、残りの実装を行い、本システムの有効性を確認する。可変長ブロックがファイルシステムの性能にどのような影響を与えるのかを調べる必要がある。またファイルの途中の変更に対するバージョン管理を行えるようにする必要がある。その場合、リンク構造について再設計を行う必要がある。

参考文献

[1] 平井貴浩,「バージョン管理機能を持つファイルシステムの設計」香川大学工学卒業論文 2002

[2] 藤本哲郎,「バージョン管理機能を持つファイルシステムのアプリケーションレベルでの実現に関する研究」香川大学工学卒業論文 2003