ファイルシステムインターフェースを利用したデータアクセスシステムの試作

4M－4

美馬 義亮、小坂 一也、根岸 康
日本アイ・ビー・エム 東京基礎研究所

1　はじめに

ファイルシステムと同様なインターフェースをもち既存のアプリケーションソフトから利用可能な、データアクセスのためのシステム Programmable File System (以下 PFS) を設計し、OS/2 Warp 上に実装したプロトタイプの評価をおこなった。[1]。本稿ではその構造と応用について報告を行なう。

2　背景

映像、音声、画像のような大量のデータを効率的に管理するためデータベースを用いることが多い。しかし、内部のバッファの存在に起因するデータアクセス時の無駄なコピーを減らし、効率的な利用を行なうには、専用プログラムの記述が必要である。

一方、マルチメディアデータには新たな形式が提案・採用されつつあり、新しい形式のデータを表示するためにはデータ形式にあわせてプログラムを更新することが要求される。多くの場合、新しい形式のデータに対応したブラウザが供給されるが、専用プログラムで表示を行なうには、プログラマに大きな負担を強いることになる。

この問題を解決するため、ファイルシステムインターフェースを持つことにより一般のアプリケーションからデータを効率良くアクセスできるシステムの設計・実装をおこなった。設計にあたって汎用性を重視したため、広い応用が可能になった。

3　構造と動作

3.1　PFS の構造

PFS は、アプリケーションからのファイルシステム呼びだしに対するインターフェースとなるファイルシステム部分と、ファイルシステムからシステムコールごとに呼び出されて実際のデータアクセスを行なうためのユーザ・プログラムからなっている。さらにファイルシステム部分はファイルシステムへの呼び出しを受けとるカーネル部分とデータアクセスを実行するデーモン・プロセスに分離される (図 1)。

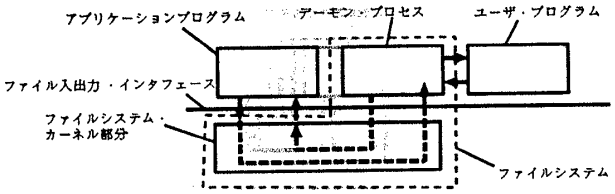

図 1: PFS の構成

ファイルシステム

OS/2 の Installable File System(IFS)[3] は、ファイルシステムの実現を容易にするためにオペレーティングシステム側が提供しているものである。IFS を用いて新たなファイルシステムを実現するためには、定められたファイル操作に対応する 30～40 個程度の関数群からなるファイルシステムドライバを作成する。

アプリケーションモード実行部分

デーモン・プロセスはアプリケーション（非カーネル）モードで実行されるプロセスである。アプリケーションプログラムによりファイルがオープンされると、ユーザ・プログラム (図 1 参照) が PFS 上のデーモン・プロセスの子プロセスとして起動され、標準入出力とパイプを用いた通信・同期を行ないながらデータのアクセスを行なう。このユーザ・プログラムには、ファイルへの Open、Close、Read、Write、Seek などに対応する動作が記述される。この方法（子プロセス＋パイプでの通信）を採用したのは、簡単にユーザ・プログラムが作れ、既存のプログラムの多くが再利用できる上に、パイプ自体がプロセス間の同期の手段としても利用できるという利点による。

A Prototyping of A Data Access System with File System Interface
Yoshiaki MIMA, Kazuya KOSAKA, Yasushi NEGISHI
IBM Research, Tokyo Research Laboratory

3.2 PFSの動作

パス名によるデータの指定

ファイルまたはディレクトリへのアクセス時に渡されるパス名は、パス名がもつシンタックス上の制限はあるものの、データを指定するために必要な情報をもたせることができる。多くの場合ユーザ・プログラムはこの情報でデータの生成や環境の設定を行なう。

データの受け渡し

ファイルのデータに対するアクセスにはランダム/シーケンシャル、読み出し/書き込み、などの違いがあり、これによって実行プログラムとのデータの管理の方法が変わる。

- ランダム読み出し：アプリケーションプログラムからランダムファイルとして読み出すためには、ファイルのオープンとともにユーザ・プログラムからの出力を、すべて臨時のファイルに書き込んだあと、あらためてこのファイルをオープンし参照する。
- シーケンシャル読み出し：人間との対話や外部プログラムとの非同期的通信を含むような場合、オープン時に仮想的ファイルの中に存在すべきすべてのデータを得ることは不可能である。このため、ユーザ・プログラムがアクセスするパイプのハンドルからデータの読み書きを行なう。
- 書き込み：書き込みはシーケンシャル読み出しと同様パイプによる通信を行なっている。これは、アプリケーションへの入力に標準入力を経由したストリームを用いているためである。

4 応用例

PFSを用いたデータアクセスの例を挙げる。PFSの使用感は「WWW上のCGIに近い」といえば実感が伝わるかも知れない。

4.1 データの自動変換・生成

ファイルの存在の問い合わせや存在しないファイルへのアクセスは、データの生成の命令として解釈することができる。「make」のようにパス名の拡張子に元ファイルの拡張子と変換プログラムを割り当てるルールをもちいれば、圧縮データの自動伸長などに利用することができる。

4.2 データベースアクセス

データベースにアクセスするためには問い合わせ言語を用いるか専用のアプリケーションを用いる必要がある。ファイルシステムにキーワードによる探索の機能を組み込んだシステムも提唱されている [2] が、このPFSを用いることにより既存のアプリケーションからデータベース中のデータを操作することが簡単に実現できるようになる。例えば、あるパス名でディレクトリを作る (移動する) ことをselect操作に対応させ、ファイル名と拡張子で最終的なデータの確定とデータの形式を指定するということが可能である。

4.3 書き込み可能CD-ROM

CD-ROMは大きなデータを配布するのに適したメディアではあるが、データの修正をすることはできない。このシステムを用いるとディレクトリアクセス操作が再定義できるので書き込み可能なファイルシステムと合わせて、二つのディレクトリを重ねあわせたように見せかけることもできる。このため、CD-ROMをあたかも書き込み可能なディスク装置のように扱うことが可能である。

5 評価

現状のアプリケーションプログラムに変更を加えず、その機能を拡張することは一般に難しいことだがPFSはそれを可能にする有力な方法の一つであるとおもわれる。

参考文献

[1] 美馬 義亮, アプリケーションの呼び出しが可能なファイルシステム信学技法, DE95-59, Vol. 95, No. 287, Oct. 1995

[2] David R. CHERITON, *UIO: A Uniform I/O System Interface for Distributed Systems* ACM Transactions on Computer Systems, Vol. 5, No. 1, Feb. 1987

[3] OS/2 File Systems Department IBM, *Installable File Systems For OS/2 Version 3.0* (IBM Internal Document)