携帯電話向けファイルシステムの開発

3C-3

阿部 大将† 小高 健二† 山口 真吾† 並木 美太郎†
† 東京農工大学工学部情報コミュニケーション工学科

1 はじめに

近年、携帯電話で JavaVM が稼動し、Java によって開発されたアプリケーションから携帯電話内臓のフラッシュメモリや SD カードなどの 2 次記憶装置が利用できるようになった。これにより有用性は向上したが、2 次記憶装置へのアクセスメソッドは携帯電話向けのプロファイルである DoJa、MIDP それぞれで異なる。これを利用するプログラマはプロファイル間で異なるアクセスメソッドを利用せざるを得ない状況にある。

そこで、2 次記憶装置を利用するプログラマから見てプロファイルごとの扱いにくさを緩和した上で、様々な記憶装置を取り扱うことを目標としたモバイル向けファイルシステム [1, 2] を実現する Java ミドルウェア「FML(File Management Library)[3]」を開発する。

2 目標

「FML」の開発の目標を、以下に示す。

(1) 携帯電話向け Java プロファイル間での 2 次記憶操作の相違を吸収する

「FML」は携帯内のストレージのほか、SD カードなどの外部のストレージもすべて同様に取り扱いできることを目標とする。また、新たな外部 2 次記憶装置が登場した場合対応をできるように拡張性を持つ設計とし、すべてのストレージをユーザプログラマが「FML」を介するで同時に取り扱えることを目標とした。

(2) 携帯電話の性能を生かす簡便なファイル管理を行う

昨今の携帯電話での JavaVM では利用できる 2 次記憶容量が上昇し、第三世代携帯電話の台頭により通信速度なども大幅に向上している。「FML」では、これらのリソースを生かすため、ディレクトリによる階層的なデータ管理やネットワークストレージをユーザプログラマに提供する。

(3) 既存の入出力クラスライブラリの下位互換とする

「FML」のアクセスメソッドは、基本的に J2SE 上でのファイルを扱う「java.io.File」などの入出力クラスライブラリのサブセットとして開発する。このことで、Java プログラマは手軽に携帯電話上の 2 次記憶装置を取り扱うことが出来るようになっている。

3 全体の構成

「FML」は、プログラマがファイルを取り扱う API クラスと、ストレージと直接データをやり取りするクラスに分割されている。

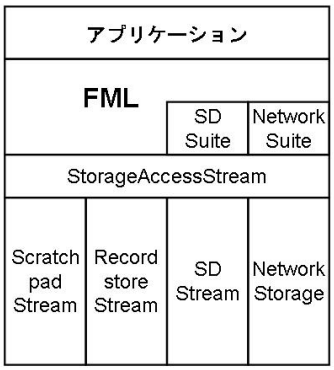

図 1: FML 概要図

API クラスは、ファイルの属性情報を定義する「File」クラス、ディレクトリの属性情報や所属ファイルを取り扱う「Directory」クラス、入出力を担う「FileInputStream」「FileOutputStream」クラス、ランダムアクセスをサポートする「RandamAccessFile」クラス、そして、これらの機能を統括する「FileManager」によって構成される。

ユーザプログラマが操作する「FML」はこれらのクラスのみとし、煩雑な管理を回避し、メモリの番地の打ち違いなどのミスを無くす。

API クラスへユーザが与えたファイルの読み書き命令は、ストレージとデータをやり取りする「StorageAccessStream」クラスに引き渡される。このクラスを窓口にして「FML」は 2 次記憶装置へアクセスできる。

Development of File System for Cellular Phones
† Daisuke Abe, Kenji Odaka , Shingo Yamaguti and Mitaro Namiki
Graduate School of Technology, Tokyo University of Agriculture and Technology

ストレージアクセスは、一定量のブロック単位で行われる。

4 FML の仕様

アプリケーションは、「FML」全体の情報を管理する「FileManager」クラスの実体を取ることでファイルを利用することができる。ユーザにはルートディレクトリのインスタンスが与えられ、これを起点としての相対パスを扱うことができる他、絶対パスを利用することもできる。

```
// ファイルマネージャを開く
FileManager fm = FileManager.open();
// ルートディレクトリの下にファイルを作る
File file = new File(fm.root,"example");
// 絶対パスで指定できる
// (file と file2 は同じファイルを指す)
File file2 = new File("/example");
// 入出力は java.io.File クラスと同様
FileInputStream fis
= new FileInputStream(file);
fis.read(buffer);
fis.close();
```

ユーザは File オブジェクトを扱うことで、ファイル、ディレクトリを操作する。

5 評価

「FML」で、ファイルサイズが一定のときのブロックのサイズによっての入出力時間と、ブロックサイズが一定のときのファイルサイズごとの入出力速度を計測した。計測を行った機種は SHARP 社の NTTDoCoMoNo 向け携帯 SH900i と、 Nokia 社の海外向け携帯電話 N-Gage である。

まず、ファイルサイズを 4096 バイトに固定し、ブロックサイズを 32 〜 512 バイトとしたときの入出力時間は次のようになった (図 2)。基本的に、ブロックサイズが大きなほうがアクセス回数が減るため入出力速度は速い。速度にばらつきは保存場所の検索により発生したものと見られる。次に、ブロックサイズを 512 バイトとし、ファイルサイズを 1024 〜 32768 バイトとしたときの入出力時間は次のようになった (図 3)。容量が最大時には大幅に時間がかかっているが、概ね SH900i の方が成績がよい。これはヒープサイズが非常に大きいためである。 N-Gage の書込時間は特に増大のはレコードストアの仕様の影響と考えられる。全体として、処理時間がかかっているのは、入出力をすべてブロック単位で実行していることなどが原因である。キャッシュなどの工夫を行う必要がある。

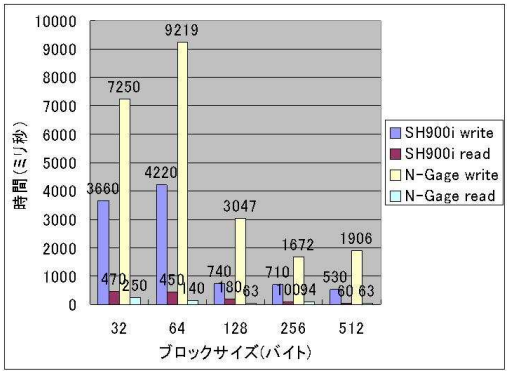

図 2: ブロックサイズごとの所要時間

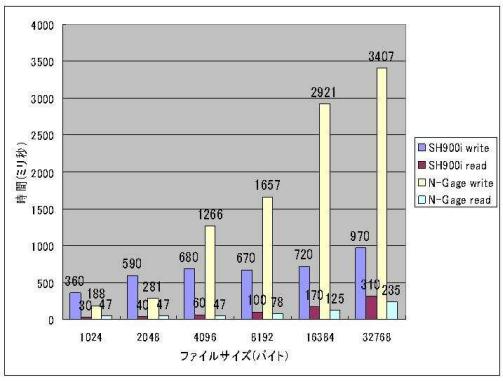

図 3: ファイルサイズごとの所要時間

6 おわりに

携帯電話 JavaVM でのファイルシステムである「FML」の設計と実装について述べた。現在は、 FML の基本的な機能を実装し、ネットワークストレージなど拡張部について構築を行っている。今後は、各種ストレージの利用を行い、効果と問題点について示して行きたい。

参考文献

[1] Carl Downing Tait, A File System for Mobile Computing, UMI Order No. GAX94-12850(1993)

[2] 石墨紀孝, 最所圭三, 福田晃, 組み込みシステム向けフラッシュメモリファイルシステムの設計, 電子情報通信学会論文誌, Vol.J84-D-I, No.1, pp.90-99(2001.1)

[3] 益子由裕, 並木美太郎, 携帯電話向けの XML 処理用ミドルウェア, 情報処理学会論文誌, Vol.45, No.SIG3(ACS5), pp.79-90 (2004.3)