7Q-8

携帯電話 Java 向けファイルシステムライブラリの開発

佐々木 悠† 小高 健二†† 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†/東京農工大学大学院工学教育部††/
東京農工大学大学院共生科学技術研究部‡

1 はじめに

現在，携帯電話上のプロセッサの処理速度の向上とともに，2 次記憶装置として利用できるメモリサイズが数 100KB まで拡張されている．しかし，現在の開発環境で携帯電話 Java 向けのアプリケーションを開発する場合，キャリアが独自に策定した Java プロファイルを利用しなければならず，入出力をはじめ，様々な処理で可搬性の低いコードを作成しなければならない．本開発では，プロファイル間の入出力の違いを吸収するライブラリを開発し，携帯電話 Java 向けアプリケーションの開発を支援していく．内部記憶装置への対応に加え，HTTP 通信を用いたネットワークストレージの利用も実現した．

2 本研究の目標

携帯電話の Java 向けファイルシステムライブラリ「FML」を開発する．日本の携帯電話向け Java プロファイルは，CLDC プロファイルを拡張した DoJa プロファイル，MIDP プロファイルを拡張した，JSCL, KDDIP が代表的である．

本開発では，CLDC プロファイル（DoJa）と，MIDP プロファイル（JSCL, KDDIP）間の入出力の違いを吸収し，プロファイルに依存しない共通した入出力環境として，J2SE の java.io 以下のサブセットを提供することを目標とする．さらに，今後の拡張性を考慮した設計とネットワークストレージのための専用サーバの試作を行う．

3 FML

FML は携帯電話上のストレージに対する統一したアクセス方式を提供するファイルシステムライブラリである．携帯電話向け Java では，ファイルシステムは搭載されておらず，記憶装置へのアクセスは，プロファイル固有の API を利用するしかない．プロファイル固有の API は一部を除いてファイルシステムは搭載されておらず，極めて低レベルな入出力を利用するしかない．そこで，FML は，UNIX のような一般的なディレクトリツリー形式のファイルシステムを提供する．プログラマは，FML をアプリケーションに組み込むことによって，プロファイルに依存しない形で，ディレクトリツリー形式のファイルシステムを利用することができる．ファイルへのアクセス方式としてシーケンシャル／ランダムアクセスを提供しており，そのインターフェースは，J2SE の java.io クラスのサブセットとし，携帯電話固有の API を知らない開発者でも，手軽に利用することができるものとした．図 1 に FML の概要図を示す．図のように FML は大きく分けて，Interface 部，

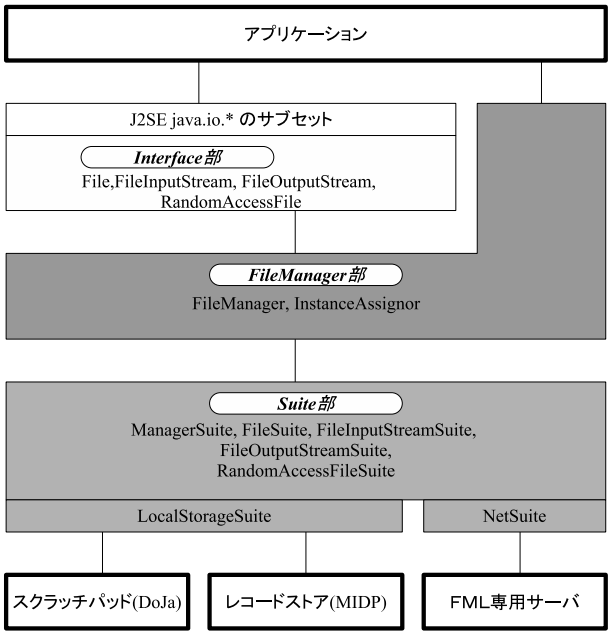

図 1: FML 全体の概要図

FileManager 部，Suite 部の 3 つの層で構成されている．プロファイルや，外部記憶装置の違いは全て Suite 部で吸収できるような構成となっている．次節でそれぞれの層の役割について述べる．

3.1 Interface 部

Interface 部はプログラマと FML の接点となる部分であり，java.io 以下のサブセットとなるインターフェースを提供する．Interface 部には，File, FileInputStream, FileOutputStream, RandomAccessFile の 4 つのクラスが存在する．これらのクラスが持っているメソッドは，J2SE の java.io 以下に存在する同名のクラスのサブセットである．

Interface 部へのアクセスで，プログラマ側に必要な情報は，そのファイルのパス名とファイル名だけであり，ディレクトリツリーの一部が外部ストレージにマウントしている場合でも，プログラマはその違いを意識することなくアクセスすることが可能となっている．

3.2 FileManager 部

FileManager 部はディレクトリツリー，マウント情報の管理を行い，Interface 部には存在しない管理メソッドをプログラマに提供する．FileManager 部に存在するクラスは FileManager と InstanceAssignor クラスである．FML を組み込んだアプリケーション上から，FML のファイルシステムを利用する際，FileManager クラスの，open メソッドを呼び出す必要がある．このメソッドが呼ばれると，FileManager は内部記憶装置からディレクトリツリーを作成し，その後，/dev/fstab に格納されているマウント情報に従って，外部ストレージにマウントをかける．

InstanceAssignor クラスは，Interface 部のクラスの

A Development of a Filesystem Library for Java Runtime Environment on Cellular Phones

† Yuh Sasaki
Department of Computer, Information and Communication Sciences, Tokyo University of Agriculture and Technology
†† Kenji Odaka
Graduate school of Engineering, Tokyo University of Agriculture and Technology
‡ Mitaro Namiki
Graduate school of Engineering, Tokyo University of Agriculture and Technology

インスタンスが作成された時に利用されるクラスである．Interface 部のクラスのインスタンスが生成される時に渡されるファイルのパス名から，そのファイルの存在するストレージを特定し，適切な Suite を選択する役割を持っている．

3.3 Suite 部

Suite 部は入出力の処理を実装している層であり，対象となる 2 次記憶装置毎に実装する必要がある．Suite 部のクラスとしては，ManagerSuite, FileSuite, FileInputSuite, FileOutputSuite, RandomAccessSuite の 5 つのクラスが存在する．使用する 2 次記憶装置に毎に，これら 5 つのクラスを継承して入出力処理の実装クラスを作成する必要がある．この層で，記憶装置ごとの違いを吸収する．

4 NetSuite

制限つき*ではあるが，携帯電話 Java から HTTP 通信を利用することができる．NetSuite とは，HTTP 通信を利用したネットワークファイルシステムを FML で利用できるようにするものである．

携帯電話の通信の性質上，頻繁に回線が切断されてしまうことが予想される．例えば携帯端末からサーバにファイルを送信している途中で，通信が切断された場合ファイルの整合性がとれなくなる恐れがある．

このようなエラー処理を考慮して NetSuite を試作した．

4.1 FML 専用サーバ

HTTP 以外のプロトコルは利用できないので，ファイルの送受信を可能にするためには，サーバ側に cgi を設置しておく必要がある．携帯電話からサーバ上の cgi へ，パラメータとデータを渡すことによって，ファイルの送受信を行う．

4.2 携帯電話からサーバへのファイル送信

前節で述べたように，ファイルの通信中に回線が切断される可能性がある．このことを考慮してサーバ側では，シャドウページ法を用いたファイルの更新を行うこととする．

表 1 にファイルの送信手順を示す．

表 1 携帯電話からサーバへのファイル送信

	携帯電話		サーバ
(1)	送信リクエスト	→	
(2)		←	該当するファイルの複製 (シャドウ) を作成 準備ＯＫ
(3)	送信データの分割		
(4)	分割データの送信	→ ←	シャドウを更新 更新ＯＫ
(5)	更新完了通知	→	シャドウで 元データを上書き

携帯電話からの送信リクエスト (1) を受け取ったサーバは，サーバ上でファイルが上書きされる場合，シャドウを生成する (2)．(3) で送信データを分割する．データの分割を行うことによって，通信回数は増えるが，エラーが発生した場合はすでに終了している通信に関しては，無駄にならずにすむ．(4) は，(3) で分割した個数回実行することになる．この時点でエラーが発生した場合，エラー発生前に送信したデータはサーバ側に残るため，エラー発生時から再び送信を行うことができる．全てのデータ送信後は (5) のコミット通知をサーバ側に送る．この通知が送られると，サーバは更新対象ファイルをシャドウで上書きする．ファイルの送信途中では，更新対象ファイルに一切変更を加えないのでデータの保全性は高められる．

4.3 携帯電話からサーバファイルの受信

上節で述べた送信と似た手順をとる．携帯電話が受信のリクエストをサーバ側に送ると，サーバではそのファイルを分割して携帯電話に送信可能状態とする．ファイルの送信する場合と同様に，複数回に分けてデータを受信する．ファイルを分割して受信することで，途中でエラーが発生した場合でもその時点からの再会が可能となっている．

5 実装と評価

本研究では，内部記憶装置向けの FML の実装，Apache 2.0, PHP 4 を用いた FML 専用サーバの試作を行った．通常の機能を持った FML のライブラリ容量は，jar ファイルで 55KB となった．DoJa プロファイルでは，アプリケーションの最大容量が 100KB であるため，かなり大きく感じるが，今後，Vodafone や KDDI に対抗してアプリケーションの最大容量が増加していくと考えられるため，大きな問題にはならないと考える．FML 専用サーバの構築は，完了しており，サーバとのファイルの送受信ができることも確認した．今後は，FML で NetSuite を実装し，Interface 部で定義されるインターフェースを提供する必要がある．

N901iS を用いて，内部記憶装置 (スクラッチパッド) への読み書きのアクセス速度を測定した．これを図 2, 図 3 に示す．

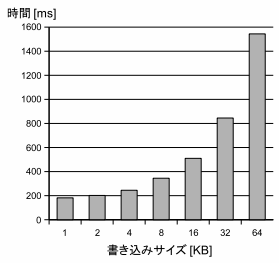

図 2: 書き込み速度

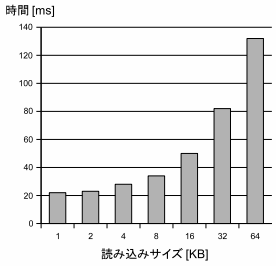

図 3:読み込み速度

書き込み速度は決して早いとはいえないが，読み書き可能な最大サイズ 400KB を書きこんだとしても，9000 ミリ程度である．現段階では，キャッシュは実装しておらず，ロジックも改良の余地があるため，今後更なる高速化は可能であると考えられる．

6 おわりに

本論文では、携帯電話 Java 向けファイルシステムライブラリ「FML」について述べた。内部記憶装置の入出力の統一化は既に可能となった．NetSuite に関しては，現時点では専用サーバとの通信しかできないが，専用サーバをプロクシとすることによって，HTTP プロトコル以外の通信を用いた，他サーバとの連携が今後の課題となってくると考えられる．

参考文献

[1] Charles Zhang and HansArno Jacobsen : Resolving Feature Convolution in Middleware Systems, ACM SIGPLAN Notices, Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages and Applications, Vol.39 , pp.188-205 (2004)
[2] 阿部 大将：携帯電話向け Java でのファイルシステムの開発 (第 67 回全国大会)
[3] 布留川英一：MIDP 2.0 携帯電話 Java アプリ開発ハンドブック
[4] アスキー編集部：i モード Java プログラミング

* DoJa では配布元のサーバにしか接続できない．また，MIDP2.0 では MIDlet に署名が必要である．