# 不揮発性メモリを用いた単一レベル記憶 OS における ファイルシステム API

2012SE126 小塚 岬
指導教員:宮澤 元

## 1 はじめに

現在フラッシュメモリのように電源を切ってもデータが消えることのない不揮発性メモリが普及している．これらは補助記憶装置として広く用いられており，例えば SSD(Solid State Drive) などのように HDD(Hard Disk Drive) に代わって大容量の不揮発性メモリが標準的なストレージデバイスとして利用されているものもある．

SSD や HDD がブロックストレージと呼ばれ，ブロック単位でしかアクセスすることができない一方，近年では，バイト単位でアクセスできる次世代不揮発性メモリの開発が進んでいる．特に，MRAM(Magnetoresistive Random Access Memory) は，現在主記憶装置として普及している揮発性の DRAM(Dynamic Random Access Memory) に匹敵する高速性と耐久性を持っている．

これらの不揮発性メモリを利用することにより，主記憶装置と補助記憶装置の両方を不揮発性メモリで構成する計算機アーキテクチャが登場するのではないかと予想される． MRAM の集積度はフラッシュメモリなどに比べると低いが，バイト単位でアクセス可能なメモリを大容量のブロックストレージと組み合わせて大容量のバイトアクセスストレージを提供するような研究もなされており [1]，このようなアーキテクチャでは，全てのオブジェクトが永続性を持ち，かつバイトアクセス可能となる単一レベル記憶を実現できると考えられる．

本研究の目的は，このようなアーキテクチャ上で動作する OS における効率的な永続オブジェクトの管理手法を提案することである．このような OS ではプロセスはファイルのような永続オブジェクトも含め，全ての情報をメモリ参照によってアクセスできるので，従来のファイルシステム API によるオーバーヘッドを削減したり，アドレスを含むようなデータ構造をそのままファイルに格納し，プロセス間で効率的に共有することが可能になる．

本稿では，このような単一レベル記憶アーキテクチャにおけるファイルシステム API を提案する．そして，ファイルシステム API の中でも特に，プロセスの仮想アドレス空間上に配置されたファイルに直接アクセスする API を提供する．実装した提案システムのプロトタイプについても述べる．

## 2 単一レベル記憶

メモリとストレージを区別せず一つのアドレス空間で管理するコンピュータシステムのアーキテクチャを単一レベル記憶と呼ぶ．主に，IBM のサーバーなどで実現されている [2]．不揮発性メモリの開発が進んできた今，必要とされる考え方の一つになると予想される．一つの巨大な仮想アドレス空間でメモリストレージとブロックストレージを管理することで ，ストレージにデータを読み書きする際に，メモリを介す必要が無いのでデータを読み書きする時間を短縮することができ，高速性が増す．ここでは単一レベル記憶を持つ代表的な OS を紹介する．

### 2.1 Opal

Opal[3] は単一レベル記憶を実現しており，64bit 仮想アドレッシングを使用する OS である．ポインタベースのデータ構造は，そのまま参照することができ，任意の時点でプログラム間で共有されるので，変換を必要とせずに二次記憶装置に直接格納することができる．

### 2.2 Multics

Multics[4] では，セグメント (ファイル) をプロセスのアドレス空間にマップしてアクセスする単一レベル記憶を実現している．

## 3 研究の概要

本研究が前提とするアーキテクチャにおけるアドレス空間の取り扱いについて述べた後，提案するファイルシステム API について説明する．

### 3.1 物理アドレス空間

単一レベル記憶でメモリとストレージのファイルオブジェクトがアドレス空間にマップされている．図 1 に本研究で想定するアーキテクチャにおける物理アドレス空間を示す．ファイルは全ての固有のアドレスを与えられ，物理アドレス空間上でアクセス可能である．

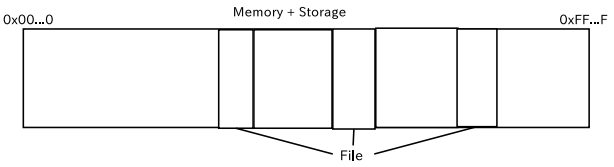


図 1 物理アドレス空間

### 3.2 仮想アドレス空間

プロセスの実行形式は 0 番地から置かれるので，プロセスごとに仮想アドレス空間を用意し実行ファイルオブジェクトは適切にアドレス変換される必要がある．図 2 に実行ファイルが仮想アドレス空間上でアドレス変換される様子を示す．実行ファイルは位置依存であり，ファイルとして割り当てられた物理アドレスから，実行可能な仮想アドレス (0 番地) に変換される．実行ファイルの以外のファイルオブジェクトは仮想アドレス空間上にマップされ，アクセスできる．ファイル単位のアクセス許可情報は，各仮想アドレス空間におけるメモリアドレスのプロテクションとして実現される．

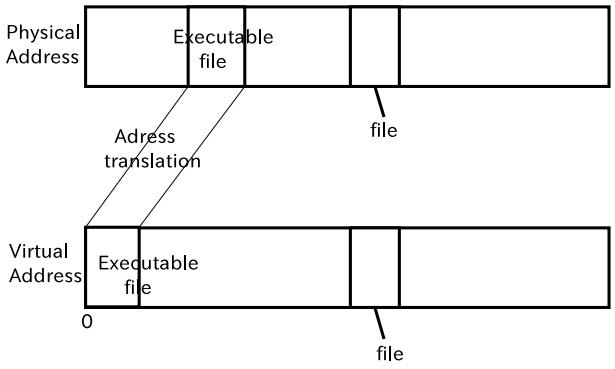


図 2 実行ファイルのアドレス変換

### 3.3 ファイルシステム API

ファイルはアドレス空間上にマップされているので，直接メモリ上でアクセスできる．ファイルにアクセスするためには，アドレス API を用いてファイル名からファイルのアドレスを知る必要がある．

アドレス API は従来のファイルシステムのパス名をそのまま利用し，パス名からアドレスを返す．パス名がわかれば，ファイルのメタデータからパーミッションを確認し権限があれば，プロセスの仮想アドレス空間上にファイルをマップする．

プロセスごとのローカル空間でファイルを共有することで，プロセス間でファイルの内容などを共有することができる．もちろん，プロセスのユーザーがそのファイルにアクセス権限を持たなければ，仮想アドレス空間上にはファイルはマップされず，ファイルにはアクセスできない．

この API によって，ファイルアクセスはメモリアクセスとして行われるので，従来の read や write などのファイルアクセス API の多くは必要なくなると考えられる．

### 3.4 不揮発性メモリ使用での技術的優位性

不揮発性メモリを主記憶装置として用いることで，各ファイルにメモリ上で固有のアドレスをより簡単に持たせることができる．揮発性メモリを主記憶として用いた場合も，固有のアドレスをメモリ上で持たせることができるが，電源を落とすとアドレス情報が消去されるので，メモリ上にマップされているファイルのアドレスの情報は，ストレージデバイスに保管して起動時にストレージ上のアドレス情報を利用してファイルをマップしなければならない．不揮発性メモリを主記憶として用いれば，その手間を省くことができる．また，固有のアドレスをファイルが持つことで，プロセス間でデータ構造を共有することが容易になる．

## 4 実装

本研究が前提とするアーキテクチャを採用する計算機は現在存在しないので，通常の PC を用いて，本ファイルシステムのプロトタイプをユーザレベルのライブラリとして実装した．

### 4.1 API

表 1 に作成した API を示す．

表 1 作成した API

| 関数名 | 内容 |
|---|---|
| kfs_init() | ルートディレクトリをマップし，k_stat 構造体にファイルの情報入れる |
| kfs_faddr() | アドレス API. 仮想アドレス空間上にマップされたファイルのアドレスを返す |

kfs_init() 関数はシステムの初期化に用いる．kfs_init() 関数によって，ルートディレクトリ情報を持つファイルをプロセスの仮想アドレス空間上にマップする．

kfs_faddr() 関数では，渡されたファイル名から所有者のユーザー ID，グループ ID，容量，アクセス保護の情報を取り出す．そして，そのプロセスのユーザー ID, グループ ID を取得し，パーミッションを確認する．その後，ファイルやプロセスのユーザー ID やグループ ID を比較し，アクセス保護を確認した後，権限があれば mmap でそのプロセスの仮想アドレス空間上にファイルをマップする．

なお，ファイルのアドレス情報をファイルのメタデータとして持たせるために，stat 構造体を拡張した k_stat 構造体を定義している．

### 4.2 サンプルプログラム

実装した API を使うことで，プロセス間でデータ構造の共有をするアプリケーション作成が変わるということを示すために，以下のプログラムを作成した．

- 二分探索木に一定時間ごとに要素を追加していくプログラム．
- 二分探索木を受け取って検索するプログラム．

## 5 まとめ

本研究では不揮発性メモリを用いた単一レベル記憶 OS におけるファイルシステム API を提案した．提案した API を持つファイルシステムのプロトタイプをユーザレベルのライブラリとして実装した．今後，システムの完成度を向上するとともに，永続オブジェクトの管理方法についてさらに検討を加える．

## 参考文献

[1] 追川 修一，ブロックストレージとの組み合わせによるメモリストレージ容量拡張手法，コンピュータシステム・シンポジウム論文集, 第 26 回 pp.63-70 (2014).

[2] 単一レベル記憶域 (SLS)，https://www-06.ibm.com/systems/jp/power/software/i/seminar/pdf/iseminar_sls.pdf (2015/11/5 アクセス)

[3] Jeff Chase，Hank Levy，Miche Baker-Harvey，Ed Lazowska，Opal: A Single Address Space System for 64-bit Archtectures，in Proceedings of the Third Workshop on Workstation Operating Systems (WWOS-III)，1992.

[4] Multics，http://www.multicians.org/ (2016/1/5 アクセス)