3U−01

タスクの再構成を支援するためのファイルシステム

結城 理憲　小林 良岳　唐野 雅樹　紅谷 順　姜 亨明　中山 健　前川 守

電気通信大学大学院 情報システム学研究科

1 はじめに

近年のネットワーク環境の広範化によって，常時稼働を前提としたサーバシステムが一般に広く運用されはじめている．そのような環境においては，プログラムやシステム自身の構成をそれらの実行を停止することなく変更できるようなシステムの必要性が増している．

そこで我々は，タスク固有の情報を管理し，周囲の環境の変化にあわせてタスクを動的に再構成することが可能なオペレーティングシステム「彩 (Aya)」を開発中である [1]．Aya の管理するタスク固有の情報には以下のようなものがある．

- タスクに含まれるモジュールやデータの構成
- 各モジュールの動作状況
- タスク固有の各種ポリシ
- タスクの使用している周辺デバイス

これらは，タスクの再構成を行う際に大変重要な情報であるが，現状ではこれらの情報を把握するには，Aya の内部構造に深く依存した方法を用いねばならない．このような方法では，各情報毎に取得するための方法に違いが発生する可能性が高い．タスクの再構成のためのモジュールの差し替えやポリシの切り替え操作に関しても，現状ではそのための専用プログラムを作成する必要がある．これらの問題は，プログラム開発者やシステム管理者にとって余計な負担となるため，そのような問題を解消するために，抽象化された統一的なインタフェースを用意する必要がある．

本稿では，このような要求に応えるために，タスク固有の情報をファイルやディレクトリとして表し，動的な再構成に必要なタスク固有情報の取得や操作をファイル操作という直観的かつ簡便な操作に抽象化することによって，

- ユーザによるタスクの再構成操作の統一
- タスクの再構成コードの簡略化
- タスクの実行状態監視の為の統一されたインタフェース

といった機能を実現するファイルシステム「綴 (Tsuzuri)」を提案する．

2 Tsuzuri の構成

Tsuzuri は，VFS[2, 3] 上の 1 ファイルシステムとして実装される．Tsuzuri 上のファイルやディレクトリはモジュールやそこに含まれる関数，あるいはメモリ管理ポリシなどのタスク固有情報の抽象であり，ディスク上に実体をもつものではない．そこで，これらファイルやディレクトリは疑似ファイルとして存在し、必要に応じてメモリ上に動的にノードが作成される．

Tsuzuri は，稼働中のタスク固有情報をディレクトリツリーとして表すための基本機能を提供するフレームワーク部と，実際にファイル操作として実行タスクのモジュール差し替えやポリシ切り替えの実現をするサブモジュール部から成る．このように，実際に操作を行う部分を切り離したのは，システムの拡張によって動的再構成を行う際に管理が必要な情報が増えたとしても柔軟に対応できるためである．

2.1 サブモジュール

サブモジュールはシステム全域に共通なグローバル情報を管理する部分と，タスク固有のプライベート情報を管理する部分とに分けられる．たとえば，ポリシ管理ではタスクが選択できるポリシのリストをグローバル情報として管理し，実際にタスク毎に選択したポリシをプライベート情報として管理する必要がある．

そこで Tsuzuri では、サブモジュールは以下の 2 つのディレクトリを管理する．

グローバル部 ：**/tsuzuri/<modname>**

プライベート部：**/tsuzuri/task/<taskid>/<modname>**

このような構成にすることにより，新たに管理する情報が増えた場合にも，柔軟かつ統一性を保ったまま対応する事ができる．

現在，Tsuzuri サブモジュールとしては，以下の 3 つがある．

- タスク構造管理モジュール
- ポリシ管理モジュール
- デバイス管理モジュール

これらのモジュールを組み込んだ時に，情報がどのようなディレクトリツリーを構成するかを図 1 に示す．

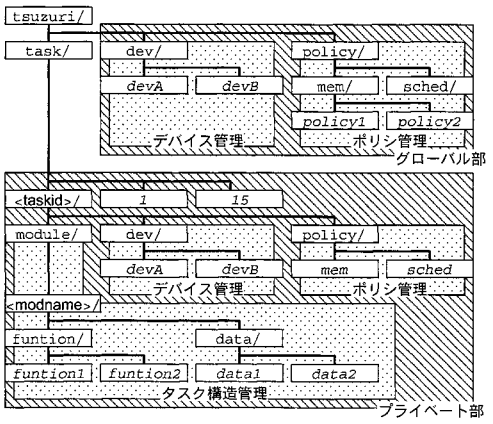

図 1: ディレクトリへの情報のマッピング

2.2 タスク構造管理モジュール

Aya では，タスクの動作状況を把握する仕組みとして Portal がある．Portal はモジュール内の個々の関数の利用状況を把握するために，カウンタ，アクセスフラグ，ジャ

A File System That Supports Run-Time Reconfiguration of System and Applications.

Yoshinori Yuuki, Yoshitake Kobayashi, Masaki Touno, Jun Beniya, Kang Hyong Myong, Ken Nakayama, Mamoru Maekawa Graduate School of Information Systems, University of Electro-Communications

ンプポインタをそれぞれ 1 つずつ保持する．これによって関数の使用状況を把握することにより，動的な差し替えを可能としている．

タスク構造管理モジュールは以下のような機能を提供する．

- ファイルを通じた Portal 情報の取得
- ファイルの複製/移動/削除による Portal 情報やモジュールの操作

Portal 情報の取得

関数毎に Tsuzuri 上に作成される疑似ファイルを利用して，モジュール差し替え時の判断材料となる重要な情報を，一般的なファイルに対するシステムコールにより読みだすことが可能になる．これにより，外部のプログラムが，あるモジュール内部の Portal の情報を取得するための専用のシステムコールを用意する必要が無くなり，取得が容易になると同時に，良くテストされた既存のコードを用いることにより，無駄なバグが混入する可能性を大幅に減らすことができる．

Portal 情報やモジュールの操作

ファイルの複製/削除/移動による操作では，ファイルあるいはディレクトリに対して一般的なファイル操作である削除や移動などを行った場合に，その操作を対応する Portal に反映する．具体的には，実際にモジュールの差し替えやタスクへの結合などの管理を担当しているシンボルマネージャの機能を内部的に呼び出す事により実現する．複製/削除/移動を行うと，ファイルシステム内部でそれぞれ以下のような操作が起きる．

複製 あるモジュールファイルを `module` ディレクトリ配下にコピーすると，コピーしたモジュールと同名のものが現在のタスクに存在するかどうかをチェックし，モジュールをメモリにロードしシンボルマネージャの機能を呼びだし，差し替えを行う．

削除 ファイルの削除を行った場合は，対応する Portal を削除する．Portal が削除されたモジュールは以後，再構成は不可能になるが，プログラムの実行は若干パフォーマンスが良くなる．

移動 ファイルの移動 (リネーム) を行った場合は，ファイル名の変更に対応して，シンボルマネージャ内で管理している Portal の名前を変更する．モジュール内部の名前を変更した場合に名前の対応を取る等に利用する．

以上のような方法により，ユーザが特別なプログラムや手順を使用することなく，cp や mv，rm と言ったような，一般的なファイル操作コマンドを使用した直観的な操作によって，実行時の Portal の一時的な名前変更や破棄，モジュールの差し替えを行う事が可能となる．再構成を行うユーザプログラムを作成する場合においても，再構成専用のシステムコールを直接利用する場合に比べ，名前のチェックなどがファイルシステム内部の操作に隠蔽されるため，再構成の為のコード記述は大幅に単純化される．

2.3 ポリシ管理モジュール

Aya にはポリシ管理機構があり，現状でメモリ管理ポリシとスケジューラポリシという 2 種類のポリシカテゴリが存在し，それぞれのカテゴリごとにタスクが固有のポリシを選択できるようになっている．

ポリシ管理モジュールでは，グローバル部配下にポリシカテゴリごとのディレクトリを作り，システムに登録されたポリシのリストをファイルとして生成する．プライベート部配下には，ポリシカテゴリ名のファイルを用意し，その実体はタスクが実際に使用しているポリシへのシンボリックリンクとして生成する．

2.4 デバイス管理モジュール

動的再構成を行う場合は，あるタスクで使用されるデバイスはそのタスク専用のスコープに存在することが好ましい．しかし，旧来のデバイス管理にタスク固有のスコープの概念を追加することは，プログラムの作成に余計な手間をかけることになり，場合によっては既存プログラムの大幅な修正などが必要になってしまう．

そこで，デバイス管理モジュールでは，プログラムからのデバイスアクセスは，グローバル部に作成したシステムグローバルなデバイスファイルに対して行えるようにし，アクセスが起こった際にファイルシステム内部でタスク固有デバイスへのアクセスへと変換する．

3 関連研究

プロセスファイルシステム (以下 procfs)[4] は，vnode/VFS アーキテクチャを利用した，プロセスのアドレス空間に関するインタフェースをファイルシステム上に構築した物である．procfs では，プロセスの状態取得やコントロールが可能になるが，本研究のように，ファイルの複製などのファイル操作によって，モジュールの差し替えなどの動的再構成を支援するようなインタフェースを実現することや，デバイスのタスク固有スコープを提供することはできない．

4 まとめ

本稿では，動的再構成可能なオペレーティングシステム Aya において，タスク固有の情報をファイルシステムのツリー構造にマッピングし，動的再構成のための統一的なインタフェースを提供するファイルシステム「Tsuzuri」について述べた．

Portal 情報/ポリシ情報/デバイス情報といった各種のタスク固有情報を，ファイルシステムのツリー構造を用いることによって，柔軟かつ統一的に扱う仕組みを提示した．そして，タスク固有の情報をファイルやディレクトリとして表すことによって，動的な再構成に必要なタスク固有情報の取得や操作をファイル操作という直観的かつ簡便な操作に抽象化し，その有効性を示した．

参考文献

[1] 小林 良岳, 前川 守. 動的再構成をサポートするための Portal 生成法とその評価. システムソフトウェアとオペレーティングシステム研究報告 No.84, 2000-OS-84, pp.107-114. May. 2000. 電子情報通信学会技術報告, CPSY2000-21, pp107-114, May 2000.

[2] S.R. Kleiman., Vnodes: An Architecture for Multiple File System Types in Sun Unix, USENIX Association: Summer Conference Proceedings, Atlanta, 1986.

[3] D. Rosenthal. Evolving the vnode interface. In Proceedings of the Summer USENIX Conference, pages 107–117, June 1990.

[4] Tom Killian, Processes as Files, USENIX Summer Conf. Proc., Salt Lake City June, 1984