

# GearsOS の Paging と Segmentation

桃原 優<sup>1</sup> 東恩納 琢偉<sup>1</sup> 河野 真治<sup>1,†1</sup>

**概要:** 現代の OS では、User Space で Page Table Entry によるメモリ管理を行える OS は少ない。本研究室ではメタレベルの処理を記述できる CbC と CbC を用いて実装する GearsOS の開発を行っている。CbC を用いることでメモリ管理などの資源管理を行えるようになるが、現在の GearsOS でのメモリ管理では単に Page Table Entry をコピーする Fork で実装している。さらに、資源管理を行える CbC で軽量のハードウェアでも動かせるように Arm のバイナリを出力する Xv6 という OS を CbC で書き直して GearsOS を開発する手法も行なっている。本論文では Xv6 を参考に GearsOS にメモリ管理を行う API を考察する。

## Implement Paging and Segmentation on GearsOS

**Abstract:** In modern OS, there are few OS that can manage memory by Page Table Entry in User Space. In this laboratory, we are developing GearsOS which is implemented using CbC and CbC that can describe meta-level processing. Using CbC enables resource management such as memory management, but with the current memory management with GearsOS, it is implemented by Fork, which simply copies Page Table Entry. In addition, CbC is used to rewrite the OS called Xv6 that outputs Arm binary so that it can run even lightweight hardware that can execute resource management, and also develops GearsOS. In this paper, we will consider the API that manages memory in GearsOS referring to Xv6.

### 1. Gears OS

本研究室では並列実行のサポートと、信頼性を保証する Gears OS の開発を行っている。従来の OS が行うメモリ管理や並列実行などは Meta レベルで処理される。メタレベルの処理を行える CbC という言語で Gears OS を実装する事で、ノーマルレベルから並列実行環境に合わせた記述ができるように設計や実装を行う。

#### 1.1 Continuation based C

本研究室では、Code Gear と Data Gear という単位でプログラムを記述する CbC と CbC を用いて実装する Gears OS の開発を行っている。Code Gear は CbC における最も基本的な処理の単位である。入力と出力を持ち、goto によって Code Gear から次の Code Gear へ遷移する事で継続的に処理を行う事によって並列処理を行うことができる。Data Gear は Gears OS におけるデータの基本的な

単位である。Input Data Gear と Output Data Gear があり、Code Gear の遷移の際に Input Data Gear を受け取り、Output Data Gear を書き出す。

#### 1.2 Meta Code Gear と Meta Data Gear

Gears OS ではメタ計算を Meta Code Gear, Meta Data Gear で表現する。CbC でもノーマルレベルとメタ計算を行うメタレベルの記述の2種類がある。この2つのレベルはプログラミング言語レベルでの変換として実現される。メタレベルへの変換は、Perl による変換スクリプトで実装している。Gears OS では、Meta Code Gear は通常の Code Gear の直前、直後に挿入され、メタ計算を実行する。Code Gear 間の継続はノーマルレベルでは図2の上のように見えるが、メタレベルでは Code Gear は図2の下のように継続を行っている。

図2に Gears OS の構成図を示す。

#### 1.3 Meta Competition

Gears OS では通常の処理を Computation, Computation のための Computation を Meta Computation として

<sup>1</sup> 琉球大学 理工学研究科 情報工学専攻  
Ryukyu University

<sup>†1</sup> 現在、琉球大学 知能情報コース 准教授  
Presently with Ryukyu University

割し、メモリとスワップ領域で Page を入れ替えて管理を行う。Segmentation では可変長の単位に分割するので、Paging と比べてメモリを無駄なく扱えることができる。近年では、Paging を採用している OS が多い。

### 3. GearsOS での Paging

Context に必要な Page Table を提供する Interface と User Space からアクセスする API が必要である。Page Table に相当するデータを Input Data Gear で受け取って変更した後、Context にあるメモリコントロールを担当する Meta Data Gear に goto してアクセスする。Meta Computation レベルで処理する事で User プログラムでも Page Table を操作することができる。Meta Computation に戻る際に、Page Table Entry のバリデーションをチェックして反映することで、他のプロセスから Page Table を書き換えられることを防ぐ。また、サンドボックスにおいて、他のプロセスが書き換えられた時にエクセプションを飛ばすようにする。

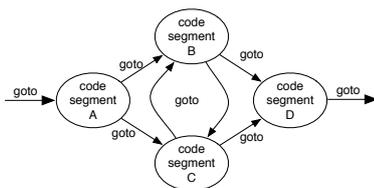


図 1 Code Gear 間の継続

#### 3.1 Segmentation の切り替え

Code Gear を切り替えるたびに Segmentation を切り替えると処理が遅くなる。Context は Thread ごとにあり、Code Gear Data Gear は自由に入れ替えることができる。しかし、Code Gear と Data Gear は煩雑に切り替わるので、Context に沿ってアクセスできる範囲を制限する必要がある。Context と Segmentation を組み合わせることで、Gears OS の Paging を実装する。

### 4. User Space で Page Table を操作する利点

現代の OS では、User Space で Page Table Entry によるメモリ管理が行える OS は少ない。User レベルの操作で Page Table が書き換えられたり、別の Page にアクセスするのを防ぐためだと考えられる。User Space でメモリ管理を行えるようにする事で、Page のバリデーションをチェックしたり、サンドボックスによる信頼性の保証が行える。また、適切な記述をすれば最適なメモリ管理を行う事ができるようになる。

### 5. Xv6 からの書き換え

#### 5.1 Xv6

Xv6 とは、マサチューセッツ工科大の大学院生向け講義の教材として使うために、UNIX V6 という OS を ANSI-C に書き換え、x86 に移植した Xv6 OS である。Xv6 は Arm のバイナリを出力するので、Raspberry Pi や 携帯など様々なハードウェアで動かすことができる。

扱う。Meta Computation の例として、並列処理の依存関係の解決や、OS が行うネットワーク管理、メモリ管理などの資源制御が挙げられる。GearsOS では Meta Computation を Meta Code Gear, Meta Data Gear で表現する。Meta Code Gear は通常の Code Gear 直後に遷移され、Meta Computation を実行する。

## 2. Paging と Segmentation

メモリ管理の手法に、Paging と Segmentation がある。Paging ではメモリを Page と呼ばれる固定長の単位に分

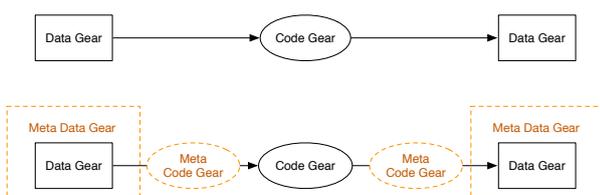


図 2 ノーマルレベルとメタレベルの継続の見え方

## 5.2 Xv6 を元にした GearsOS の実装

Gears OS はメモリ管理や CPU の管理を User Space から操作できることを目標に実装しているため、メモリに限りのあるハードウェアで動くように実装できる方が好ましい。ANSI-C で書かれている Xv6 を CbC に書き直し、それを元に Gears OS を実装していく。Xv6 の新しい要素として Gears OS の Context の部分を User Space 側で実行する。

## 5.3 表題・著者名等

謝辞 CbC や Gears OS の説明では、先行研究を元に本稿を作成した。また、河野 真治 准教授にさまざまなご教授を頂いた事を感謝する。

## 参考文献

- [1] 宮城光希 : Gears OS のモジュール化と並列 API, OS 研究会 (2018).