プログラミング言語 Perl による オペレーティングシステム構成法の研究

浅野 一成† 並木 美太郎‡

† 東京農工大学工学府情報工学科 ‡ 東京農工大学共生科学技術研究院システム情報科学部門

1 はじめに

近年，ソフトウェアの開発効率を重視する傾向から，スクリプト言語が注目されている．スクリプト言語は，動的型付けやメモリ回収機構，インタプリタや仮想マシンを用いた中間言語コードの実行を特徴し，C 言語や C++ で起こっていたメモリ関連のバグからプログラマを解放する．本報告では，スクリプト言語 Perl[1](以下 Perl) による OS「PerlOS」の構成法を述べる．PerlOS では，Perl の言語処理系 (以下 Perl 処理系) を OS が搭載されていないベアマシン上で動作させ，元来 C 言語や C++ で行っていた計算機資源管理を Perl で実現する．

2 先行事例と関連研究

Perl による単一言語システムを目標としたシステムの先行事例には Perl/Linux[2] が挙げられる．Perl/Linux は，Linux ディストリビューションとして位置づけられている．ユーザプログラムは，すべて Perl で実現されており，シェルや UNIX 互換のユーティリティ群などが作成された．しかしながら，カーネルには Linux が利用されており，カーネルを含めると，ソフトウェア階層の大半は C 言語で構成されることになる．PerlOS では，OS を含めソフトウェア階層のほとんどの部分を Perl で記述する．

Perl 以外では，Lisp，C#などで OS の構成が実現されている．Movitz[4] は Lisp で記述される OS である．AOT コンパイルを行うこと，Lisp で記述された OS を構成している．ターゲット CPU の機械語を生成するコンパイラ前提とする OS のため，CPU が変わるごとにコンパイラを新たに作る必要が生じる．CooS[5] は C#で記述される OS である．CLI を OS の実行環境として持つ．外部ランタイムプログラムに頼らないシステム構成になっているものの，言語処理系を独自実装しており，OS コードを安定させて動作させるまでの手間が生じる．

3 目標と設計方針

PerlOS では，最終的な目標として以下に挙げる資源管理を Perl を用いて記述することを目標とする．

- 割込み処理
- デバイスドライバ

A Study of Operating System by Programming Language Perl

ASANO Kazunari†, NAMIKI Mitarou‡

†‡Tokyo University of Agriculture and Technology

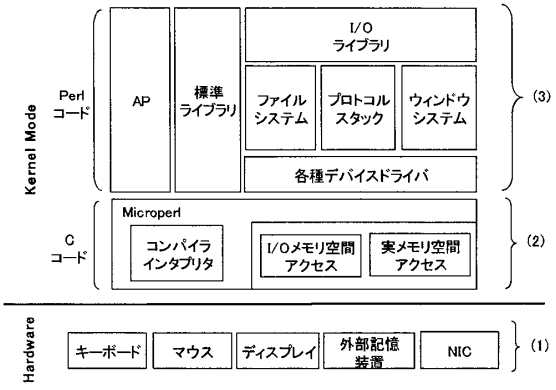

図 1 システムの全体構成

- スレッド
- ファイルシステム
- TCP/IP プロトコルスタック
- ウィンドウシステム

上記に挙げた資源管理を実現するためには，ハードウェアアクセスを伴うため，レジスタアクセスや実メモリ，I/O アクセスが生じる．そのような Perl の言語仕様に無い低レベル操作は，言語処理系の拡張機能を用いて実現する．しかしながらすべてを拡張機能によって実現することは Perl で OS を記述する目的に反する．そのため，拡張機能の利用は必要最低限にとどめる．Perl 処理系は独自実装せず，実用レベルの既存の C 言語実装の処理系を用いる．

4 全体構成と設計

本研究が目標とするシステムの全体構成を図 1 に示す．本システムは，(1) から (3) の 3 つの部分から構成される．

(1) ハードウェア

周辺装置として，キーボード・マウス・ディスプレイ・ディスク・NIC を扱う．

(2)C 言語

この階層のソフトウェアは C 言語で記述される．Perl 処理系，言語処理系の拡張機能，外部割込み処理の一部，プロセスのコンテキストスイッチの一部で C 言語が用いられる．

外部割込みは Perl 処理系に対し通達される．一旦，割込み要求を Perl 処理系内部で保留させた後，適切なタイミングで Perl で記述された割込みハンドラを起動させる．割込みハンドラは適切な Perl のデバイスドラ

イバを呼び出し，割込みを処理する．

(3)Perl

この階層のソフトウェアはすべて Perl で記述される．Perl で記述されるソフトウェアは，OS と APP の 2 つである．両者とも C 言語階層にある同一の言語処理系で実行される．

PerlOS におけるプロセスは，1 インタプリタインスタンスで実行するプログラムと定義される．言語処理系は，1 つのプログラムごとにインタプリタインスタンスと呼ばれるインタプリタの実行実体をメモリ上に生成し，その上で Perl プログラムを実行する．

OS 機能を持つインタプリンタインスタンスは特にカーネルインスタンスと呼ばれる．カーネルインスタンスはメモリ上にただ 1 つだけ存在し，計算機資源管理を行う．APP が実行されるインタプリタインスタンスは AP インスタンスと呼ばれる．複数のインタプリタインスタンスが同一メモリ上で生成された場合，個々のインスタンスの内容は独立する．APP は OS の資源管理機能を破壊しないように，限定されたオペコードのみが許されたサンドボックス内でプログラムが実行される．

プログラム間のデータ共有はスレッドを用いて行う．PerlOS におけるスレッドはネイティブスレッド方式で Perl レベルのスレッドを提供する．Perl のスレッドはインタプリタスレッドと呼ばれる．スレッドはスタックのみが分離したプロセスで表現される．

5 プログラミング例

PerlOS における資源管理をディスクドライバを例に挙げて示す．図 2 に擬似コードを示す．コードはデバイスドライバと割込みハンドラの 2 つに大別される．

デバイスドライバではメモリと I/O ポートアクセスのための XSUB が利用される．

割込みハンドラはドライバ名をキーとしてシグナルハッシュに登録する．外部割込みは Perl 処理系に通達され，処理系は割込み要因から適切な Perl のハンドラを実行する．

6 実装

PerlOS はシミュレータと実機上で開発されている．シミュレータとして VirtualPC と VMware Player を，実機として Intel Celeron の (1.7GHz/, メモリ 1GB) を用いている．Perl 処理系として Microperl[3] を開発マシン上でコンパイルし，ベアマシン上に搭載した．Microperl には，言語仕様拡張のための XSUB を用いて，I/O ポートとメモリアクセスを実現した．XSUB の実行速度を表 1 に示す．

表 1 XSUB の実行速度 [μ 秒]

	byte	word	long
in	1.94	2.47	3.48
out	1.86	3.10	6.08
read	2.53	2.42	2.69
write	1.84	1.91	1.73

```
package DISK_A;
use PerlOS::DD::PIC;
# シグナルハッシュによる割込みハンドラの登録
$SIG{HWI_DISK_A} = \&handler;
# デバイスコントロールのためのサブルーチン
sub send_command
{
   for my $cmd (@cmd)
   { outb ($cmd, ...); while (inb(...) != XXX) {;} }
   # 外部割込み待ち
   while ($intr_flag) {;}
   $intr_flag = 1;
}
# 割込み発生時呼び出されるサブルーチン
sub handler
{
   # レジスタ類は C 言語階層で退避済み
   # マスク処理と EOI 発行
   $pic->disable(DISK_A)->eoi;
   $self->disable;
   # バッファへのデータの書き出し
   while (inb(...) != XXX)
   { writeb (inb(...), $buf_addr + $i); $i++; }
   $intr_flag = 0;
   # マスク解除
   $pic->enable(DISK_A);
   $self->enable;
}
```

図 2 XSUB を用いたディスクドライバの例

XSUB を利用し，Perl プログラムでデバイスドライバ (フロッピーディスク，ハードディスク，キーボード/マウス)，FAT と ext2 ファイルシステム，プロセス管理の実装を行った．メモリ管理には CPU のセグメンテーション機構を利用し，プロセスごとに固定割付けでメモリ割り当てを行う方式を採用した．1 セグメント内のメモリ管理は，C 言語ライブラリの malloc 関数によって行われる．

7 おわりに

本報告では Perl による OS「PerlOS」の構成法について述べた．現在までに，AP レベルで Perl の組込み関数によるプロセスの並行処理を実現した．また，Perl によるプロセス管理，デバイスドライバ，ファイルシステムを実現した．今後は，標準ライブラリ threads によるネイティブスレッド方式のインタプリタスレッドを PerlOS に実現することが課題として挙げられる．

参考文献

[1] The Perl Directory, http://www.perl.org/

[2] Don Mahurin:Perl/Linux, http://perllinux.sourceforge.net/

[3] Simon Cozens:Microperl;TPJ,Volume 5, Number 3 (#19), Fall 2000

[4] Frode V. Fjeld : Movitz,Workshop on Evolution and Reuse of Language Specifications (ECOOP 2004)

[5] 高橋明生, 加藤和彦: CIL で表現された OS カーネルの実行方法, 情報処理学会研究報告 2006-OS-101