uClinux の H8/300 アーキテクチャ移植
小規模CPUでのlinuxカーネルの実装

１　背景

uClinuxはMMUを必要としない小規模なCPUで動作するlinuxカーネルである。
H8/300用のuClinuxもすでに取り込まれており海外でも知られる様になっている。
この環境はターゲットハードウエアが低価格で入手出来るので、標準のカーネルで対応することでより広く使われる事が期待できる。
また、このような環境で本格的なOSが動作することで、従来のRTOSでは難しい応用例が生まれることも期待できる。

２　目的

本プロジェクトでは、以下の二点に付いて開発を進める。
第一に最新開発版であるバージョン2.5のlinuxカーネルをH8/300へ対応させる。
MMUの無い環境に対応するための変更はすでに取り込まれているので、アーキテクチャに依存する部分と各種デバイスドライバの開発を行う。
第二にメモリ容量の制限を緩和しプログラムの起動時間を短縮するため、
toolchainに位置独立なコードを生成する機能を追加することで、共有ライブラリなどの機能を利用出来るようにする。
開発成果は極力上流へ還元し成果を広く利用してもらう。

３　開発の内容

3.1　アーキテクチャ依存部

linuxカーネルにおいて特定のCPU/ターゲットに依存する部分を抽象化する部分である。
・CPUの初期化
・例外／割り込み処理エントリ
・低レベルプロセス管理
・低レベルメモリ管理
などの処理を作成する必要がある。
また、H8/300版に固有な機能としてCPUに内蔵されている汎用I/Oポートを管理するための機構を追加している。
この部分については、uClinux-2.4.20のコードを元にして2.5カーネルでの変更を加えていく形をとった。
これは、i386とm68knommuでの修正を検討した結果基本的な構造はほぼ同じであったため、
1.既に実績のあるコードを使うことで開発にかかる時間を短縮する。

2.2.4 カーネルについても開発が継続しているので、コードを共通化していた方がバグ修正などの点で都合がいい。
など、あえて新規に開発するだけのメリットを見出せなかったためである。

3.2　2.5 カーネルでの修正部分

以下の部分について修正を行った

・context switch

1.　スケジューラーより渡される情報が変更されたため、それに対応した。

・exception handling

1.　プロセス管理情報の構造が変更されたため、それに対応した。
2.　割り込みベクタテーブルを、動的に生成するように変更した。
3.　CPU 毎に持っていたシステムコールのエントリテーブルを共通にした。

2,3 は保守性の向上のためである。

これらの開発成果は正式に取り込まれて配布されている。

3.2 デバイスドライバ

2.4 カーネルでサポートしているハードウエアを同様にサポートすることを目標にした。
具体的には以下のドライバである。

・内蔵シリアル I/F
・Ethernet
・ CompactFlash(IDE)

各ドライバは、既に他アーキテクチャ用に用意されている物をH8/300ターゲットに移植する形をとった。
具体的にはハードウエアにアクセスする部分をH8/300用に修正し、割り込みベクタ番号を修正するだけで対応出来た。
ハードウエアアクセス部分は、カーネル内部でi386の ISA バスに見せかけることでドライバを無変更で使えるようにすることも不可能ではないが、変換のオーバーヘッドや、コードサイズの肥大化などのデメリットがはるかに大きいので、ドライバ側を修正した。
割り込みに関しては割り込みコントローラーの制御がアーキテクチャ依存部の担当であり、ドライバが割り込みコントローラを意識することはないので、アーキテクチャ依存部をi386と同様の動作にすることで、ドライバ側の修正を最小限に抑えることができた。

シリアル I/F のドライバについては修正点を開発元に還元し、共通のソースコードとして配布されるようになっている。

それ以外のドライバについても、同様の状況にするため作業を継続している。

3.3 toolchain の位置独立コード(PIC)対応

3.3.1 現状の問題点

nommu の環境ではアドレス変換が出来ないため、各プロセスに割り当てられるメモリ空間がロード時に決定される。
しかし、現状の toolchain では特定のアドレスに配置されるバイナリしか生成出来ないので、プロセスの起動時にアドレスに依存する部分をロードされたアドレスに合わせて書き換えることで、任意のアドレスへのロードを実現している。
この方法では、書き換える箇所の量が多い場合プロセスの起動に非常に時間がかかってしまう。
PIC のバイナリであればこの処理を省くことができ、また CPU のメモリ空間に配置された ROM/RAM 上にファイルシステムを構築している場合は、ファイルシステム上にあるコードを直接実行する事が可能になり、メモリ消費量・起動時間ともに劇的に改善することができる。
また、もうひとつのメモリ削減作である共有ライブラリの実装にも PIC バイナリに対応した toolchain が要求される。

3.3.2 toolchain の対応

gcc についてはいくつかのマクロを定義し、必要なコードを生成するmdを追加することで PIC に対応したコードを生成するようになる。
また uClinux の共有ライブラリ機構が通常の PIC バイナリを拡張した形で実装されているので、それに対応する必要がある。
binutils(as/ld)については、 ELF の再配置タイプに PIC の属性を追加し、 GOT を取り扱えるように拡張する必要がある。
この作業については当初 gcc-3.3/binutils-2.13 に対応するものを開発していたが、H8/300版 gcc を開発している平田氏より最新版の gcc-3.4 に対応するパッチが提供されたので、将来性を考えてこちらを採用し作業を進めた。

3.3.3 カーネル及びライブラリの対応

生成された PIC のバイナリを実行するための機能をカーネルに追加し、 PIC 及び共有ライブラリ機構に対応するためライブラリを修正した。
これら実験は十分に安定している uClinux-2.4 カーネルを利用して行った。なおカーネルに対して行った修正は 2.6 でも同様の修正で対応できる。

これら作業の結果、共有ライブラリを利用したテストプログラムの動作を確認しており、コードサイズの削減など当初の期待どおりの結果を得ることができた。

４　従来の技術との相違

このクラスのCPUでPOSIXに準拠したOSを実装した例は皆無であり、またlinuxカーネルへの貢献の一例としても基調な物と考えている。

５　期待される効果

手軽に扱える組込みlinux環境としてこれ以上のものは無く、また本プロジェクトの成果である最新カーネルへの対応や共有ライブラリなど実用化するため二必要な機能もかなり整備されたので、広く組込みlinuxが使われるための環境を整備する一助になれると考える。

６　普及の見通し

成果は全て公開し、積極的に上流へ還元し取り込んでもらうことで、特に意識することなく利用できるようにしていく。
また、現状不足しているドキュメント類を整備し、より多くの人に興味を持ってもらえる様にしたい。

７　開発者名

※　佐藤　嘉則　（ysato@users.sourceforge.jp）

参考URL

http://www.kernel.org
http://www.uclinux.org
http://uclinux-h8.sourceforge.jp