Gears OSのデバイスドライバの開発
Gears OS device driver development

学籍番号:175701G 氏名: 奥田光希　指導教員 : 河野真治

概 要

An OS must be reliable and extensible. We are designing Gears OS with the goal of guaranteeing reliability for normal level calculations and scalability for meta-level calculations.Currently, It need to connect a Mac to run Geas OS on a Raspberry Pi via serial communication to get input.Being able to use a keyboard and mouse on the Gears OS on the Raspberry Pi will improve convenience. It would also eliminate the need to connect to hardware other than the Raspberry Pi through a PC.The purpose of this study is to develop a Gears OS Device Driver in CbC on a Rasberry Pi.

1　研究目的

OS には信頼性が保証できることと拡張性があることが求められている。信頼性をノーマ ルレベルの計算に対して保証し、拡張性をメタレベルの計算で実現することを目標に Gears OS を設計中である。現在,Geas OS を Raspberry Pi 上で動かすためには Mac とシリアル通信で繋げなければ入力ができない。Raspberry Pi 上の Gears OS でキーボードやマウスを使えるようになれば利便性が向上する。また、Raspberry Pi 以外のハードウェアで動かす時にも、PC を介して接続しなくて良くなる。本研究では、Rasberry Pi 上で Gears OS の Device Driver を CbC で開発しすることが目的である。

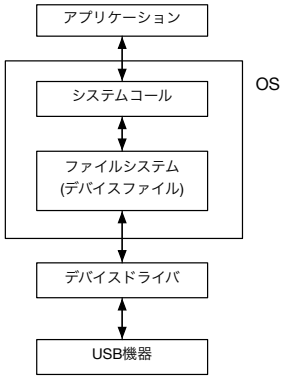

図 1: Device Driver の役割

2　Device Driver

OS は、接続された機器を直接理解することはできず、OS と接続機器の橋渡しの役割を担うのが Device Driver である。Device Driver は OS ごとに作成する必要がある。当研究室で開発されている Geas OS に対応する Device Driver として USB 接続機器が市場に多いことや Raspberry PI に接続端子があることから USB Driver を開発する。また、開発された Device Driver の信頼性の検証をしたいため、USBDriver のソースコードを CbC で書いていく。

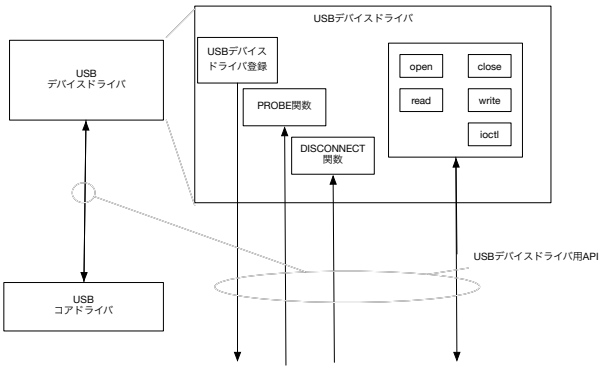

図 2: USB Driver の構成

3 Countinuation based C(CbC)

Countinuation based C(CbC) とは、当研究室で開発されているプログラミング言語である。CbC は、C からサブルーチンコールとループ制御構造を取り除き、継続を導入した C の下位言語である。CbC は Code Segment を基本的な処理単位とする。C の関数とは異なり返り値を持たないが、Code Segment の宣言は C の関数の構文と同じように行い、型に__code を用いる。CbC は for 文や while 文といったループ制御構文を持たないので、ループ処理は自分自身への再帰的な継続を行う事で実現する。 現在の Code Segment から次の Code Segment への移動は goto の後に Code Segment 名と引数を並べて記述する。この goto による処理の遷移を継続と呼ぶ。C と異なり、戻り値を持たない Code Segment ではスタックに値を積んで行く必要が無くスタックは変更されない。このようなスタッ クに値を積まない継続を軽量継続と呼ぶ。この軽量継続により、並列化、ループ制御、関数コールとスタックの操作を意識した最適化がソースコードレベルで行えるようになる。

4 Geas OS

Gears OS は当研究室で開発を行っている OS である。Gears OS の実装には GCC 上に実装した CbC を用いている。Gears OS では、プログラムの単位として Gear を用いる。Gear は並列実行の単位、データの分割、Gear 間の接続等になる。Code Gear はプログラムの処理そのものであり、任意の数の Data Gear を参照し、処理が完了すると任意の数の Data Gear に書き込む。Code Gear は接続された Data Gear 以外にアクセスできない。Code Segment と同じように Code Gear から次の Code Gear への処理の移動は goto の後に Code Gear の名前と引数を指定する事で実 現できる。Data Gear はデータそのものを表す。int や文字列などの Primitive Data Type を持っている。Gear の特 徴として処理やデータの構造が Code Gear、Data Gear に閉じている事にある。これにより、実行時間、メモリ使用 量などを予測可能なものにすることができる。

5 Raspberry Pi 上の Gears OS

先行研究で Raspberry Pi 上で Gears OS を実装するために、Raspberry Pi 上で CbC の make を行った。しかし、メモリが小さくて make するのに時間がかかる。また Raspbian では、qemu によるメモる拡張ができないので別の OS で Raspberry Pi で動く CbC を実装する必要がある。そこで Cross Compile を行うことで別の OS で Raspberry Pi で動く CbC を実装した。

5.1 Cross Compile

Cross Compile とは、別の OS で実行可能なコードを生成するコンパイル手法である。Raspbian 以外の以外の OS 環境であらかじめ Raspberry Pi 上で CbC が動くように Cross Compile を行い、そのコードを Raspberry Pi に移すことで、実行できるようになる。

5.2 xv6

xv6 とは MIT の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 Raspberry Pi に移植することができ、ANSI-C で書かれている xv6 を CbC に書き直すことで、Raspberry Pi 上で CbC を動かすことができる。

6 今後の予定

6.1 現状

現段階では、Raspberry Pi 上に Gears OS を搭載している。また、Raspberry Pi と Mac をシリアル通信で繋げることができた。これにより、Raspberry Pi 上の xv6 で CbC を書くことができる。

6.2 研究計画

今後の計画として本格的に Device Driver を開発していく。USB Driver を開発するために Raspberry Pi の UEFI から USB Controller の設定を書き換えて USB Port を所得できるようにする。次に USB の Device Name と Device Type を得られるようにし、USBCommand も所得していく。最後に TTY Driver と接続する。その後、USB Driver のソースコードを CbC に書き換えていく。

参考文献

[1] 桃原優, 河野真治.Gears OS on Raspberry Pi(2018)

[2] 坂本昂弘, 河野真治.xv6 kernel 上での CbC による interface の実装 (2019)

[3] 桃原優, 坂本昂弘, 河野真治. 継続を用いた x.v6 kernel の書き換え (2019)

[4] 宮城光希, 河野真治,Code Gear と Data Gear を持つ Gears OS の設計 (2018)

[5] 自作 OS の今と昔.https://knowledge.sakura.ad.jp/22042/

[6] Russ Cox,Frans Kaashoek,Robert Morris. xv6

[7] 福谷武司, 小谷章二, 高橋智.Linux による USB デバイスドライバ作成と制御インタフェース開発

[8] 城戸英之, 吉田泰彦, 大原茂之.Linux 用 USB デバイスドライバの開発支援に関する一提案 (情報処理学会第 67 回全国大会,2005)

[9] デバイスドライバ開発入門.https://prev.net-newbie.com/linux/driver/index.html