

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 Raspberry 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、Raspberry Pi 上で Gears OS の Device Driver を CbC で開発しすることが目的である。

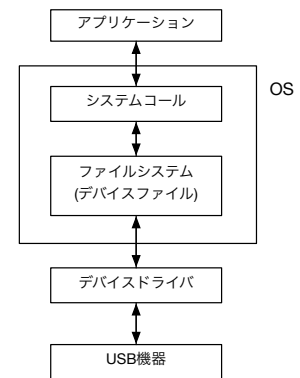


図 1: Device Driver の役割

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 (図 1)。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。USB Driver の構成は図 2 のようになる。また、開発された Device Driver の信頼性の検証をしたいため、USB Driver のソースコードを CbmakC で書いていく。

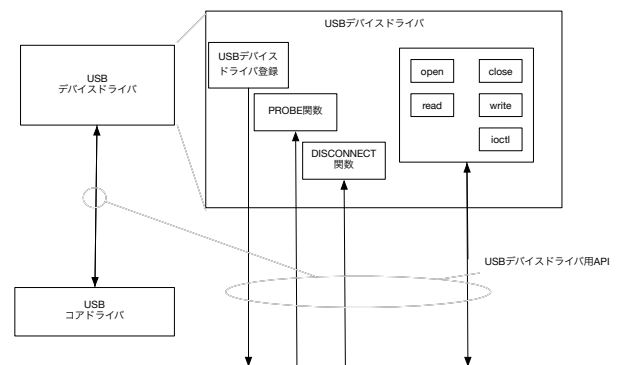


図 2: USB Driver の構成

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の関数とは異なり戻り値を持たないが、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によるメモリの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、実行できるようにする。

5.2 xv6

xv6とはMITのオペレーティングコースの教育目的で2006年に開発されたオペレーティングシステムである。xv6はオリジナルであるv6が非常に古いC言語で書かれている為、ANSI-Cに書き換えられx86に再実装された。xv6はreadやwriteなどのsystemcall、プロセス、仮想メモリ、カーネルとユーザーの分離、割り込み、ファイルシステムなどUnixの基本的な構造を持っている。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をbootできるようにする。その後、xHCIを実装するために、そのUSBプロトコルをCbCで実装していきたい。また、CbCのバージョンが上がったのでCross Compilerも新たに作成する必要がある。

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>